初心者からプロまで必携

パソコンPCシリーズ 8001 6001 ハンドブック

朝日新聞電子計算室編

朝日新聞社

800円



は じ め に

世はあげてパソコン時代である。しかし，あまりの急成長のためか，パソコンのベストセラーであるNECのＰＣ－８００１ですら，まとまった機能解説のハンドブックがなかった。朝日新聞社のマイコン研修会でも，教育機器の観点から使用機種にＰＣ－８００１を選んだが，発足後半年ほどたったころ，会員の間から自然発生的に「使いやすく，適切なハンドブックが欲しい」という声が出てきた。たしかに，機能・操作の各項目について断片的な資料は出ているけれども，まとまったものはない。そこで，会員のためにハンドブックをまとめよう，ということになった。

約一年間かけて編集が終わりかけたころ，出版局プロジェクト室から，せっかくまとめたのだから単に社内用の資料に終わらせず，一般に公開して，広く社会に役立ててはどうかとのすすめがあった。「それでは初心者からプロまで，だれにでも役立つようなハンドブックを作ろう」と意気込み，新しく発売されたＰＣシリーズの普及版，ＰＣ－６００１の機能解説も加えて，再編集したのがこのハンドブックである。初心者がどこで行き詰まるかも，マイコン研修会の研修記録から洗い出して内容に反映させた。初心者研修用のテキスト例は，自習用としても役立つはずである。ＰＣ－６００１については巻末にまとめておいた。

責任上，本文・表を通じて島戸が執筆した。さらに江田，西村，杉山を加え，徹底的な討議を繰り返して改稿した。ＮＥＣ技術陣からも，多数の資料と有益な意見をいただいた。

このハンドブックが，パソコンを使いこなし，機能を最大限にひき出す上でお役に立てば幸いである。

朝日新聞社電子計算室長

島戸　一臣

本書の特長・使用法

（1）コマンドおよびステートメントの説明は次の原則にしたがう。

<　>で囲まれたものは使用者が指定する。

〔　〕で囲まれたものは必要なら指定する。

（ディスク）とあるのは，ディスクが接続されているときに使える。

（2）16進表示は次の原則にしたがう。

ＢＡＳＩＣの場合は数字の頭部に＆Ｈをつける。

機械語やニーモニックの場合は数字の末尾にＨをつける。

（3）索引はコマンドやステートメントだけではなく，機能でも引けるようにした。

（4）15回分のテキストは，社内研修で約400人が使ったものに，加筆訂正を加えて作った。あらかじめキーボードに慣れさせておかないと，受講者のレベルにもよるが1回2時間では苦しいかもしれない。

◇　この本は執筆者4人がワードプロセッサーを使い，直接，版下を作りました。

◇　恐れいりますが，この本の内容に関するお問い合わせは，ハガキか封書でお願いいたします。ご返事は電話でいたしますので，電話番号を忘れずにお書き添えください。

あて先　〒104　東京都中央区築地5－3－2

朝日新聞社　出版局プロジェクト室

執筆者

島戸　一臣　　１９３６年生まれ

朝日新聞社電子計算室長兼技術本部システム管理室長

朝日新聞社マイクロコンピューター研修会事務局長

江田　総司　　１９５３年生まれ

朝日新聞社電子計算室員，システムリーダー

第一種情報処理技術者

西村　　哲　　１９５７年生まれ

朝日新聞社電子計算室員，第一種情報処理技術者

杉山　裕一　　１９５８年生まれ

朝日新聞社電子計算室員，第一種情報処理技術者

表紙デザイン　東　盛太郎

目　次

目　次

PC−8001 キー配置図

ファンクションキー　SHIFT+f1→TIME＄　SHIFT+f2→KEY　SHIFT+f3→PRINT
の定義 ──────→f1のみ──→HT　f2のみ──→auto　f3のみ──→goto

ファンクションキー→

f・1	f・2	f・3	f・4

! 1 ヌ	” 2 フ	＃ 3 ァ ア	＄ 4 ゥ ウ	% 5 ェ エ	& 6 ォ オ	’ 7 ャ ヤ	(8 ュ ユ	) 9 ョ ヨ

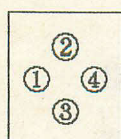
ESC	Q タ	W テ	E イ	R ス	T カ	Y ン	U ナ	I ニ

CTRL	A チ	S ト	D シ	F ハ	G キ	H ク	J マ	K ノ

SHIFT	Z ッ ツ	X サ	C ソ	V ヒ	B コ	N ミ	M モ

GRAPH	スペースキー

（1）キー



① キーを押すと，①の位置の文字が出る。英字の場合は小文字。
② SHIFT を押してキーを押すと②の位置の文字が出る。
英字の場合は大文字が出る。
③ カナキーを押してキーを押すと③の位置の文字が出る。
④ カナキーとSHIFT を押してキーを押すと④の位置の文字が出る。

（2）ESCキー
プログラムの一時停止。再開はキーボード上のどのキーを押してもよい。

（3）CTRLキー
スクリーンエディットのときに使う。このキーを押しながら次のキーを押す。

Ｒ→ＩＮＳキーと同じ。
Ｈ→ＤＥＬキーと同じ。
Ｎ→次の単語にカーソルを移す。
Ｂ→前の単語にカーソルを移す。
Ｊ→カーソル以後を次の行に分割する。
Ｅ→カーソルから行の終わりまでを消す。
Ｇ→ＢＥＥＰ
Ｃ→入力待ちのときＳＴＯＰと同じ。
Ｉ→水平タブ。
Ｋ→ＨＯＭＥキーと同じ。
Ｌ→ＣＬＲキーと同じ。
Ｍ→ＲＥＴＵＲＮキーと同じ。

（4）ＳＴＯＰキー

プログラムの強制停止。再開はＣＯＮＴコマンド（ＣＯＮＴ参照）

SHIFT+f4→list. SHIFT+f5→cont+RET
f4のみ──→list　f5のみ──→run+RET

f・4	f・5				
	0 ヲ ワ	＝ － ホ	＾ へ	￥ ー	ＳＴＯＰ
Ｏ ラ	Ｐ セ	＠ 〃	［ 「 。	ＲＥＴＵＲＮ	
	Ｌ リ	＋ ； レ	＊ ： ケ	］ 」 ム	カ ナ
＜ ， ネ	＞ ． 。 ル	？ ／ ・ メ	＿ ロ	ＳＨＩＦＴ	

（5）CLR →押すと画面クリアー。
HOME→SHIFT+CLR カーソルが左上へ。
DEL →カーソルの左１文字を消して
　　ひとつ詰める。
INS →カーソルの位置に空白を挿入。
矢印→カーソルを各方向に移動。

HOME CLR	↓ ↑	← →	INS DEL
7	8	9	＊
4	5	6	＋
1	2	3	＝
0	，	．	RET

（6）ＲＥＴＵＲＮキー
入力の終わりに必ず押すこと。

（7）スペースキー
このキーを押すと空白がひとつとられる。

（8）カナキー
押すとロックされてカナモード になる。解除はもう一度押す。

				年	月	日	時	分	秒	▊	▊	円	STOP
	╭	╮	◢	◣	／	＼	╳	♠	♥	▊	▌		RETURN
	╰	╯		◥	◤	●	○	◆	♣	▊	▌	▔	
	▁	▂	▃	▄	▅	▆	▇	█	▏	▕			

┌	┬	┐	─
╞	╪	╡	═
├	┼	┤	│
└	┴	┘	RET

GRAPH →
キー

上の文字および図形を出すには，GRAPH キーをおしながら各キーを押す。

ＢＡＳＩＣの基本文法

(1) ＢＡＳＩＣ言語で書かれた処理手順の命令をステートメントという。ステートメントの組み合わせでプログラムを作る。プログラムへの指令をコマンドという。ステートメントのなかにはコマンドとして働くものもある。

(2) プログラムは小文字で入力しても大文字で入力してもかまわない。

(3) ダイレクトモードでは電卓のように使える。この場合，行番号をつけない。

```
PRINT 3.14159*45
? SQR(3)
FOR J=1 TO 10:PRINT J*2;:NEXT J
```

プログラムモードでは，まずプログラムとして記憶する。それから実行する。したがって，なん回でも実行できる。行の始めに必ず行番号がくる。

```
10 INPUT R
20 A=3.14159*R
30 PRINT A
```

(4) 行番号は0から65529までの正整数でつける。1行の長さは255文字まで。行番号や空白も1字に数えられる。コロン（:）で区切ると1行に複数のステートメントが書ける。（これをマルチステートメントという）

(5) 定数とはＢＡＳＩＣの実行時，そのまま使える値をいう。数値定数と文字定数とがある。文字定数は引用符（”　”）で囲む。

（6）変数とはＢＡＳＩＣプログラムのなかで使われる値を表す名前をいう。数値変数と文字変数とがある。変数のアタマ１文字だけは英字とする。変数のなかに予約語（ＰＲＩＮＴ，ＩＦ，ＯＮなど）や特殊文字（”や：など）を含めてはいけない。変数の長さは２文字以内。２文字以上つけるとアタマ２文字で判定される。文字変数は，語尾に＄（ドルマーク）をつける。

（7）式とは，定数，変数，関数，文字，演算子の組み合わせで，結果が数値か文字列になるものをいう。

（8）数値と文字列は別々に扱う。文字列は”　”で囲うこと。

（9）演算の優先順位は

カッコで囲まれた式→べき乗→負数→＊，／→￥→ＭＯＤ→＋，－→条件判定→ＮＯＴ→ＡＮＤ→ＯＲ→ＸＯＲ→ＩＭＰ→ＥＱＶ

操作上の注意点

（1）スイッチを切って再度入れるときには，５秒以上待つこと。

（2）リセットボタンを押すと初期状態にもどる。プログラムも消える。

ＳＴＯＰボタンを押しながらリセットボタンを押すとプログラムは消えずに，画面がクリアーされて　ＯＫ　と出る。

（3）初心者はＲＥＴＵＲＮキーを押し忘れる。前のプログラムをＮＥＷで消さないで次のプログラムを入力することがある。

文字・演算子

	文　　　字	意　　　味
機能	&Oまたは&	8進定数　&はアンパーサントと読む
	&H	16進定数　&H51など
	疑問符（?）	PRINTと同じ　L?とは使えない
	アポストロフィ（’）	REMと同じ　1行の途中からでも使える
	ピリオド（.）	現在の行番号　たとえばIllegal Function Call in 30と出た場合にはLIST.で30行が出てくる
	コロン（：）	1行に複数の文を入れるときの区切り記号
	セミコロン（；）	PRINT文での区切り記号
	コンマ（，）	PRINT文を14字ずつ区切る
	引用符（”）	引用符で囲むと文字列になる　”ABC”や”6”
型宣言	$	ストリング（文字列）を宣言する　A$やB$など
	%	−32768から32767の整数宣言　A%など
	!	7．1桁の浮動小数の単精度実数宣言　A!など
	#	16．8桁の浮動小数の倍精度実数宣言　B#など
算術演算	=	代入または等価判定　C=A+Bなど
	−	負数表示または減算　−15やA−Bなど
	+	加算や文字列の結合。A+Bや”AB”+”CD”
	*	乗算　A*Bや3*2，あるいはC*4など
	／	結果が浮動小数点である除算　A／BやC／2
	^	べき乗　A^2　これはA*Aと同じ
	¥	結果が整数である除算　11¥3は3
	MOD	結果が整数である剰余　11MOD3は2

文字・演算子

	文字	意味
論理演算	NOT	否定または1の補数　結果は整数
	AND	論理積　結果は整数
	OR	論理和　結果は整数
	XOR	排他的論理和　どちらか一方にビットが立っていれば1，両方立っていれば0　両方とも0は0
	EQV	等価　XORの否定になる　整数
	IMP	インプリケーション　Xの否定またはY　X IMP YはYが1なら1　あとはXの否定　整数
関係演算	=, >, < <>, >< =<, >= =>, <=　など	条件判定　整数，真ならー1，偽なら0になる

コマンドとステートメント

n,X,Y,Z は数値
S は文字列とする

一般	コマンド	AUTO CONT DELETE KEYLIST LIST LLIST MON NEW RENUM RUN TERM
	ステートメント	CLEAR DATA DEFFN DEFDBL DEFINT DEFSNG DEFSTR DIM END ERASE FOR GOSUB GOTO IF~THEN LET NEXT ON~GOSUB ON~GOTO READ REM RESTORE RETURN STOP SWAP DEFUSR
	入出力ステートメント	INPUT LINEINPUT LPRINT PRINTUSING LPRINTUSING OUT POKE PRINT WAIT
	画面ステートメント	COLOR CONSOLE GET@ LINE LOCATE PRESET PSET PUT@ WIDTH
	特殊ステートメント	BEEP ERROR KEY ON ERROR~GOTO TROFF TRON RESUME
	数値関数	ABS (X) ATN (X) CDBL (X) CINT (X) COS (X) CSNG (X) EXP (X) FIX (X) INT (X) LOG (X) RND (X) SGN (X) SIN (X) SQR (X) TAN (X)
	ストリング	ASC (S) CHR$ (X) HEX$ (X) INKEY$ INPUT$ (X) INSTR ([X,] S1, S2) LEFT$ (S, X) LEN (S) MID$ (S, X, Y) OCT$ (X) RIGHT$ (S, X) SPACE$ (X) STR$ (X) STRING$ (X, S又はY) VAL (S)
	その他の関数	CSRLIN DATE$ ERL ERR FRE (X) INP (X) LPOS (X) PEEK (X) POINT (X, Y) POS (X) SPC (X) TAB (X) TIME$ USRn (X) VARPTR

コマンドとステートメント

n,X,Y,Z は数値
S は文字列とする

ディスク	コマンド	FILES FORMAT LFILES LOAD MERGE MOUNT NAME REMOVE SAVE SET
	ステートメント	FIELD LSET RSET
	入出力ステートメント	CLOSE GET# INPUT# KILL LINEINPUT# OPEN PRINT# PRINT#USING PUT#
	ストリング	INPUT# (X〔,〔n〕〕)
	ディスク関数	CVD (S) CVI (S) CVS (S) DSKI$ (X, Y, Z) EOF (n) FPOS (n) LOC (n) LOF (n) MKD$ (X) MKI$ (X) MKS$ (X) ATTR$ (X) DSKF (n)
カセットテープ	コマンド	CSAVE CLOAD CLOAD?
	入出力ステートメント	INPUT#−1 PRINT#−1
	特殊ステートメント	MOTOR

PC−8001のN−BASICはマイクロソフト社のM−BASICを土台にしている。さらに図形に関する命令などが追加されている。

大部分のマイクロコンピューターがM−BASICを土台にしているので，命令も機能も似かよったものが多い。

＜　＞で囲まれたものは使用者が指定する。
〔　〕で囲まれたものは必要なら指定する。
（ディスク）とあるのは，ディスクが接続されているときに使える。

（1）　コマンド

ＡＵＴＯ〔〔＜開始行＞〕，〔＜間隔＞〕〕

ＡＵＴＯ（ＲＥＴＵＲＮキーを押す。以下同じ）で自動的に行番号を発生させる。停止はＳＴＯＰキーを押す。

ＡＵＴＯ…………………１０番から１０の間隔で１０，２０，３０と発生させる。
ＡＵＴＯ100，100‥‥100，200，300………………
ＡＵＴＯ，５……………１０，１５，２０，２５………………

ＣＬＯＡＤ”＜ファイル名＞”

カセットテープからプログラムをロードする。

ＣＬＯＡＤ”ＡＢＣ”………ＡＢＣと名前のついたプログラムをロードする。
ＣＬＯＡＤ”トリタテ”……トリタテと名前のついたプログラムをロードする。

ＣＬＯＡＤ？”＜ファイル名＞”

ＣＳＡＶＥ後，正しくセーブ（記録）されたかどうか検証する。
メモリーの中身は消えない。

ＣＯＮＴ

ＳＴＯＰで停止したプログラムの実行を再開する。ステートメントのＳＴＯＰ参照。

ＣＳＡＶＥ”＜ファイル名＞”

プログラムをカセットテープにセーブ（記録）する。

ＤＥＬＥＴＥ〔＜開始行＞〕〔－＜終了行＞〕

指定した行を抹消する。

ＤＥＬＥＴＥ　３０　…………………３０行を抹消。３０と入力して　ＲＥＴキーを押したのと同じ。
ＤＥＬＥＴＥ　１００－１５０　………１００行から１５０行を抹消。
ＤＥＬＥＴＥ　－５０…………………５０行までを抹消。

（ディスク）ＦＩＬＥＳ〔＜ドライブ番号＞〕

ディスクに存在するファイル名と大きさを画面に一覧表示する。ドライブ番号を指定しなければ１番が指定される。.

（ディスク）ＦＯＲＭＡＴ　〔＜ドライブ番号＞〕

ディスクをフォーマッティングして，使用可能にする。

第一レベルのフォーマット
ＦＯＲＭＡＴステートメントを実行する。これでディスク（ＰＣ－８０３１）で読み書きできる。
ＢＡＳＩＣからもＤＳＫＩ＄，ＤＳＫＯ＄で入出力ができる。他のコマンドやステートメントは使えない。

第二レベルのフォーマット
ＰＣ－８０３１付属のシステムディスクにあるｆｏｒｍａｔプログラムを使う。ＦＡＴ，ディレクトリー，ＩＤの初期化ができる。これでＢＡＳＩＣのコマンドやステートメントが実行できる。

ＫＥＹＬＩＳＴ

ファンクションキーのｆ・１からｆ・１０の内容を画面に表示する。

（ディスク）ＬＦＩＬＥＳ

プリンターへＦＩＬＥＳを出力する。

ＬＩＳＴ〔＜開始行＞〕〔－〔＜終了行＞〕〕

プログラムを画面表示する。

ＬＩＳＴ…………………全部表示
ＬＩＳＴ．－………………該当行からを表示。
ＬＩＳＴ－．………………該当行までを表示。
ＬＩＳＴ４００……………４００行のみを表示。
ＬＩＳＴ４００－…………４００行からを表示。
ＬＩＳＴ－４００…………４００行までを表示。
ＬＩＳＴ１００－４００……１００行から４００行を表示。
ＬＩＳＴ．…………………エラーになったときなどの該当行を表示。

ＬＬＩＳＴ〔＜開始行＞〕〔－〔＜終了行＞〕〕

プリンターへプログラムを出力する。使いかたはＬＩＳＴと同じ。

（ディスク）LOAD”<ファイル名>”

プログラムをディスクからメモリーへロード（転送）する。

下の手順でロードできる。

```
MOUNT1
LOAD"トリタテ"
REMOVE
```

（ディスク）LOAD”<ファイル名>”，R

それまで開いていたデータファイルはそのままで，指定したプログラムをロードして実行する。大きなプログラムを分割して実行するときなどに使う。



BBがロードされると，AAもそこで使っていた変数も消える。データの受け渡しはディスクのデータファイルを仲介させる。

（ディスク）MERGE”<ファイル名>”

メモリー上にあるプログラムに対して，ディスクにあるファイルをまぜ合せる。同じ行番号は，ディスクファイルのプログラムと置きかわる。ファイルはアスキー形式（SAVE”<ファイル名>”，A）でセーブされたものであること。

MON

機械語モニターに制御を移す。アスタリスク（＊）が出る。モニター・モードからBASIC・モードに戻るには，CTRLキーとBを押す。

モニター・モードで使えるコマンドは次のとおり。

＊SXXXX	セット・メモリー。メモリーに機械語を書き込む。XXXXは16進の番地。
＊DXXXX，YYYY	ダンプ・メモリー。XXXX番地からYYYY番地の内容を画面表示する。
＊L	ロード・テープ。テープから機械語を読み込む。終了すると＊が出る。
＊WXXXX，YYYY	XXXX番地からYYYY番地の機械語をテープへセーブする。
＊LV	セーブした内容をベリファイする。
＊GXXXX	XXXX番地へジャンプする。機械語だけのプログラムの場合は，これで動き出す。
＊TM	メモリーのテスト。
＊CTRLキー　＋　B	リターンベーシック。モニターからBASICに戻る。

（ディスク）MOUNT〔＜ドライブ番号＞〔，＜ドライブ番号＞‥‥〕〕

ディスクを設定する。MOUNTだけならすべてのドライブが対象となる。

（ディスク）NAME”＜旧ファイル名＞”AS”＜新ファイル名＞”

ファイル名を変更する。

NEW

メモリー上のプログラムと変数をクリアーする。

RENUM〔〔＜新行番号＞〕〔〔，＜旧行番号＞〕〔，＜間隔＞〕〕〕

行番号をつけ直す。GOTO文の行き先も自動的にかわる。

RENUM‥‥‥‥‥‥‥‥‥‥‥‥‥10，20，30‥‥‥‥とつけ直す。
RENUM100，35，20‥‥‥‥旧35行を100行として，以降20行間隔につける。
RENUM100，，50‥‥‥‥‥‥‥100，150，200‥‥‥‥とつけ直す。

（ディスク）REMOVE〔＜ドライブ番号＞〔，＜ドライブ番号＞‥‥〕〕

MOUNTしてあるディスクを抜きとる前に，FAT（58ページ）の書き込みをやり，抜きとってもいい状態にする。ライト・プロテクト・ノッチの銀紙がはりつけられてある場合には，REMOVEは不要である。

RUN〔＜行番号＞〕

プログラムを実行する。行番号を指定すると，そこから実行する。

（ディスク）RUN”＜ファイル名＞”〔，R〕

ディスクからファイルをロードし実行する。実行前に開いているファイルは閉じて，メモリーはクリアーする。Rをつけるとデータファイルは開いたまま。

（ディスク）SAVE”＜ファイル名＞”〔，A〕

プログラムをディスクへ保管する。Aをつけると，アスキー・コード（キャラクターコード表のコード）でプログラムをセーブする。つけなければ，バイナリー形式になり圧縮してセーブされる。REMOVEを絶対に忘れないこと。

下の手順でセーブできる。

```
MOUNT1
SAVE"トリタテ"
REMOVE
```

（ディスク）ＳＥＴ〔＜ドライブ番号または#ファイル番号または”ファイル名”＞，〔＜属性文字＞〕〕

ファイルの属性をきめる。
属性文字は

Ｒ	リード・アフターライト。書いたあと読む。
Ｐ	ライトプロテクト。書き込み禁止。

ＳＥＴ1，”Ｒ”…………1番ドライブへの出力は，Ｒ検査をする。ただし，ディスクがマウントされている間だけ。
ＳＥＴ#1，”Ｒ”………ファイル1が開かれている間，そこへの出力はＲになる。
ＳＥＴ”ＡＢＣ”，”Ｐ”………ファイル”ＡＢＣ”を削除・変更から保護。
ＳＥＴ1，””……………ドライブ1のすべての属性をなくす。

ＴＥＲＭ＜コード＞，＜パリティー＞，＜ボーレート＞，＜フィールドスイッチ＞

ターミナルモードにする。ターミナルモード参照。

コード	Ａ	アスキーコード
	Ｊ	ＪＩＳコード
パリティー	0	ＮＯＮ
	1	ＯＤＤ
	2	ＥＶＥＮ
ボーレート	0	基準周波数×６４
	1	基準周波数×１６
フィールドスイッチ	0	ＯＦＦ
	1	ＯＮ………キャリッジリターンコードを受信すると自動改行。

（2）　ステートメント

ＣＬＥＡＲ〔〔＜文字領域の大きさ＞〕，＜メモリーの上限＞〕

変数を0，文字変数をヌルストリング（””）にクリアーして，文字領域の大きさ（初期値は３００）とＢＡＳＩＣが使えるメモリーの上限を指定。
ＣＬＥＡＲ５００，＆ＨＤＦＦＦ………文字領域を５００に，メモリー上限を＆ＨＤＦＦＦ番地にする。次の＆ＨＥ０００番地から，機械語を入れることができる。

ＤＡＴＡ＜定数＞〔，＜定数＞………〕

ＲＥＡＤ文で読まれる数値や文字を割り当てる。ＲＥＡＤ文と対で使う。

```
READ  A, B, C
READ  A$, B$, C$
  ・
  ・
DATA  1, 5, 7, X, Y, Z
```

DEFFNx（＜引数の並び＞）

ユーザー関数を定義する。

```
10 DEFFNA(R)=3.14*R^2
20 X=FNA(5)
30 Y=FNA(10)
40 PRINT X;Y
```

関数名をAとした。変数Rは10行のなかだけで対応していればよい。

DEFUSR〔＜数字＞〕=＜番地＞

機械語サブルーチンの開始番地を定める。数字は0から9まで。与えなければ0。

```
10 DEFUSR2=&HE000
       ・
       ・
80 Y=USR2(X)
```

BASICでXを入力すると，機械語サブルーチンが計算して，結果をYにもどす。BASICとサブルーチンの仲立ちはレジスターがする。ユーザールーチン参照。

DEFINT＜変数名＞〔＜－変数名＞〕
DEFSNG＜変数名＞〔＜－変数名＞〕
DEFDBL＜変数名＞〔＜－変数名＞〕
DEFSTR＜変数名＞〔＜－変数名＞〕

指定した変数の型を上から整数，単精度，倍精度，文字型に定義する。ただし，型宣言文字（%，!，#，$）が，つねに定義に優先する。

DEFINT A－E
A，B，C，D，Eの文字で始まる変数を整数型に定義する。
DEFDBL F
文字Fで始まる変数を倍精度に定義する。

DIM＜変数名＞（＜要素の数＞〔，＜要素の数＞…………〕）

配列の大きさを決めて，メモリーに領域を確保する。

DIM A(2)….

A(0)	A(1)	A(2)

まで3個を確保。

DIM X(5)….

X(0)	X(1)	………	X(5)

まで6個を確保。

テキスト第6回，第7回の例を参照。

DIM Y（2，3）……2次元で確保する。次のようになる。

	0	1	2	3
0	Y（0，0）	Y（0，1）	Y（0，2）	Y（0，3）
1	Y（1，0）	Y（1，1）	Y（1，2）	Y（1，3）
2	Y（2，0）	Y（2，1）	Y（2，2）	Y（2，3）

END

プログラムの実行を終了する。

ERASE<配列名>〔，<配列名>………〕

配列を抹消する。したがって，そのメモリースペースを別用途に使える。
DIMA（x）などを抹消せずにDIMA（y）などと再宣言するとRedimensioned arrayエラーになる。

（ディスク）FIELD〔#〕<ファイル番号>，<フイールドの大きさ>AS<文字変数>…………

ランダムファイルバッファの中に変数用領域を割り当てる。ファイル番号は，そのファイルを開いたときに指定したもの。フイールドの大きさは，文字変数に割り当てるバイト数。文字変数がCで始まるときはASとの間にスペースを入れること。

```
FIELD1，10ASA$，20ASB$，50AS C$
```

ランダムバッファの最初の10バイトをA$に，次の20バイトをB$に，次の50バイトをC$に割り当てる。合計で255バイトまで割り当てられる。FIELD文の変数名をINPUT文やLET文で使わないこと。使うと文字領域と混ざりあって，ランダムバッファの位置を正しく指さなくなる。

FOR<変数名>=<式>TO<式>〔STEP<式>〕

NEXTまでの処理を条件によってくりかえす。STEP<式>は，増加（減少）の幅。なければ1ずつ増加する。FOR～NEXTと対で使う。

```
10 FOR K=0 TO 10 STEP 2
20 PRINT K;
30 NEXT K
```

上のプログラムは実行すると，0　2　4　6　8　10と結果が画面表示される。
また，10 FOR K =N TO 10 として，N>10の場合でも1回は実行する。

```
FOR  A=････ ─────┐
FOR  B=････ ──┐  │
   ･          │  │
NEXT  B ──────┘  │
NEXT  A ─────────┘
```

多重ループの場合，内側のＮＥＸＴ文は，外側のＮＥＸＴ文の前にあること。

ＧＯＳＵＢ＜行番号＞

指定した行からはじまるサブルーチンへとぶ。サブルーチンの最終行にはＲＥＴＵＲＮがあること。

ＧＯＴＯ＜行番号＞

指定した行へプログラムの制御を移す。

ＩＦ＜条件式＞ＴＨＥＮ＜文または行番号＞〔ＥＬＳＥ＜文または行番号＞〕

条件式が真ならＴＨＥＮのあとの命令を，偽ならＥＬＳＥのあとの命令を実行する。ＴＨＥＮとＥＬＳＥの関係は，ＥＬＳＥからみた左側のＴＨＥＮで他のどのＥＬＳＥにも対応していないＴＨＥＮとの組になる。

ＬＥＴ＜変数名＞＝＜式＞

変数に値を代入する。ＬＥＴは省略できる。

```
LET  A=3.14159*R^2
```

（ディスク）ＬＳＥＴ＜文字変数＞＝＜文字表記＞

ＰＵＴ文の準備として文字領域をメモリーからランダムバッファに左づめで移す。数値は，ランダムバッファに移す前に文字にしなくてはならない。ＬＳＥＴ（ＲＳＥＴ）を実行する前に，ＦＩＥＬＤ文があること。

```
LSET  A$="HAPPY"
LSET  B$=MKS$(B)
LSET  C$=X$
```

Ａ$，Ｂ$，Ｃ$はＦＩＥＬＤ文で定義されていること。Ａ$が１０バイトとってあれば，ＨＡＰＰＹ　　　　　と入る。

ＮＥＸＴ〔＜ループ変数＞〔，＜ループ変数＞〕〕

ＦＯＲループの範囲を決める。ＦＯＲ参照のこと。

ON<式>GOSUB<行番号>〔, <行番号>‥‥〕

式の値に対応して，サブルーチンへとぶ。

```
ON A-1 GOSUB 100, 200, 500
```

A－1の値が1なら100行，2なら200行，3なら500行へ。その他の値だと次の行へ制御がうつる。ただし，条件の式の値が負だとエラー。

ON<式>GOTO<行番号>〔, <行番号>‥‥〕

式の値に対応した行番号へとぶ。

READ<変数>〔, <変数>‥‥〕

DATA文のデータを各変数に読みこむ。DATA参照。

REM〔<任意の文字列>〕

プログラム中に注釈をつける。アポストロフィー（ ’ ）でもおなじ。マルチステートメントにはできない。

```
10  REM  **KEISAN**
20  REM………空白の1行ができる。
50  ' ** ジュンビ **
60  PRINT A 'ケッカノプリント
```

RESTORE〔<行番号>〕

DATA文はREADで1回読むと，あとは使えない。それを解除して，DATA文をはじめから読めるようにする。あるいは，指定した行のDATA文から読めるようにする。したがって，解除して読む場合，READ文の前にはRESTORE文が必ずあるはず。

RETURN

サブルーチンから，そのサブルーチンを呼びだしたGOSUB文の次の行へもどる。したがって，サブルーチンの最後の文はRETURN。

```
50   GOSUB 200
60   PRINT A
     .
200  A=25*X
300  RETURN
```

200行，300行がサブルーチン。最後にRETURNがある。

(ディスク) ＲＳＥＴ＜文字変数＞＝＜文字表記＞

ＬＳＥＴ参照。右づめで移す。

ＳＴＯＰ

プログラムの実行を中止してブレークメッセージを画面表示，コマンドモードへもどる。ＥＮＤ文とちがって，使用中のファイルを閉じないので，デバック中などにダイレクトモードで中間結果を調べたり，変更したりできる。ＣＯＮＴまたはＧＯＴＯ行番号で再開できる。プログラムを変えるとＣＯＮＴはきかない。

ＳＷＡＰ＜変数＞，＜変数＞

ふたつの変数の値を交換する。変数の型は一致していること。

```
１０　Ａ＝３，Ｂ＝４
２０　ＳＷＡＰ　Ａ，Ｂ
３０　ＰＲＩＮＴ　Ａ，Ｂ………Ａ＝４，Ｂ＝３になる。
```

(３)　入出力ステートメント

(ディスク) ＣＬＯＳＥ〔〔＃〕＜ファイル番号＞〔，〔＃〕＜ファイル番号＞……〕〕

ディスクファイルへの入出力を終了する。
ＣＬＯＳＥだけだと，オープン状態のすべてのファイルをクローズする。ＥＮＤ文とＮＥＷコマンドでもそうなる。ＳＴＯＰでは，ディスクファイルのクローズはやらない。

(ディスク) ＤＳＫＯ＄＜ドライブ番号＞，＜トラック番号＞，＜セクター番号＞

ドライブ，トラック，セクターを指定して，そのセクターにＦＩＥＬＤ＃０で指定したランダムバッファの内容を書きこむ。ＭＯＵＮＴコマンドなしで動いてしまう。したがって，ファイル・アロケーション・テーブルやディレクトリーと無関係に動くので危険。ＤＳＫＯ＄はステートメントだが，逆の機能を持つディスク関数にＤＳＫＩ＄がある。

(ディスク) ＧＥＴ＃＜ファイル番号＞〔，＜レコード番号＞〕

ランダムディスクファイルからランダムバッファに１レコード読みこみ，同時にＦＩＥＬＤ文で指定した各フィールドに対応した変数にそのデータを代入する。レコード番号は最大５４４まで。省略すると最後にＧＥＴした次のレコードが，バッファに読みこまれる。

ＩＮＰＵＴ〔”＜プロンプト文＞”；〕＜変数＞〔，＜変数＞‥‥〕

キーボードから入力する。変数は数値変数名でも添字つき変数名でも文字変数でもよい。文字列の入力は，引用符で囲む必要はない。

ＩＮＰＵＴ　”ツギノデータ”；Ａ

（ディスク）ＩＮＰＵＴ＃＜ファイル番号＞，＜変数名＞〔，＜変数名＞‥‥〕

ディスクのシーケンシャルデータファイルからデータを読み，プログラムの変数に代入する。ファイル番号が－１のときは，カセットからの入力。

（ディスク）ＫＩＬＬ”＜ファイル名＞”

ディスク上から指定したファイルを削除。

ＬＩＮＥＩＮＰＵＴ〔”＜プロンプト文＞”，〕＜文字変数＞

特殊文字を含めて１行全体（２５５文字以内）を区切らずに入力する。

（ディスク）ＬＩＮＥＩＮＰＵＴ＃＜ファイル番号＞，＜文字変数＞

１行全体を区切らずにシーケンシャルデータファイルから文字変数に読む。

ＬＰＲＩＮＴ＜式＞〔，または；＜式＞‥‥‥‥〕

プリンターへ出力。ＰＲＩＮＴ参照。

ＬＰＲＩＮＴ　ＵＳＩＮＧ”＜フォーマット文＞”；＜式＞〔，＜式＞‥‥〕

プリンターへ出力。ＰＲＩＮＴ　ＵＳＩＮＧ参照。

（ディスク）ＯＰＥＮ”＜ファイル名＞”〔ＦＯＲ＜モード＞〕ＡＳ〔＃〕＜ファイル番号＞

データの入出力対象となるディスクファイルの使用宣言。

モード　　ＩＮＰＵＴ‥‥‥‥入力ファイルの指定。ファイルポインターは，そのファイルの先頭を指す。ファイルがないと

エラーになる。
OUTPUT……出力を行う。ファイルがあれば，その先頭をファイルポインターは指す。ファイルがなければ新しいファイルを作り，ファイルポインターはその先頭を指す。
APPEND……追加。そのファイルの最後のレコードの直後をファイルポインターは指す。
省略すると………ファイルがあれば，その先頭をファイルポインターは指す。なければ，そのファイル名のファイルを作り，ファイルポインターはその先頭を指す。

OUT<ポート番号>，<データ>

出力ポート（0～255の整数）にデータ（0～255の整数）を送る。

```
OUT  &H40, &H20……………BEEP1
OUT  &H40, &H00……………BEEP0
```

POKE<メモリー番地>，<データ>

メモリー内の指定した番地にデータを書き込む。メモリー番地は，&H0000（10進数の0）から&HFFFF（10進数の65535）の範囲，データは&H00（0）から&HFF（255）までの範囲であること。

PRINT<式>〔，または；<式>……….〕

画面表示する式は数値や文字でもかまわない。式を省略すると改行される。
コンマ（，）で区切ると，14字ずつ区切られる。セミコロン（；）で区切ると，次の値は連続してプリントされる。数値のプリントは，頭に符号の入るスペース，末尾にスペース1個がとられる。文字列は引用符（" "）で囲むこと。式の最後にコンマもセミコロンもなければ，改行される。キーワードPRINTの代りに疑問符（?）が使える。LPRINTの代りにL？は認められない。引用符（" "）とセミコロン（；）が隣接するときには，セミコロンを省略できる。

PRINT USING<"フォーマット文">；<式のリスト>

文字や数値を指定したフォーマットでプリントする。PRINT USINGの一覧表を参照のこと。

（ディスク）PRINT#<ファイル番号>，〔USING<フォーマット文>；〕〔<式のリスト>〕

シーケンシャルファイルにデータを書く。ファイル番号は，ファイルがオープンされたとき指定されたもの。－1だとカセットテープへの出力になる。フォーマット文はPRINTUSINGのフォーマットと同じ。式のリストを複数書くときにはセミコロン

（；）で区切ること。カンマ（，）で区切ると余分な空白も書きこむ。

```
A$="TOKYOTO"
B$="CHIYODAKU"
PRINT#1，A$；B$
```

はTOKYOTOCHIYODAKUとディスクに書かれる。

```
PRINT#1，A$；"，"；B$
```

とするとTOKYOTO，CHIYODAKUとなり，２つの文字変数としてふたたび読める。

（ディスク）PUT#<ファイル番号>〔，<レコード番号>〕

ランダムディスクファイルにランダムバッファから１レコードを書き込む。レコード番号が省略されると，PUT文で最後に書いた次の番号が割り当てられる。レコード番号の上限は５４４。

WAIT<ポート番号>，<マスク>，<論理スイッチ>

入力ポートをモニターする。　（ポートから読みこんだデータ　XOR　論理スイッチ）AND　マスク　が真ならプログラムの実行を再開する。３２～３３ページ参照。

（４）　スクリーンステートメント

COLOR，LINE，PSETなどでつかわれるファンクションコードのアトリビュート（属性）

コード	白黒モード	カラーモード
0	ノーマル	黒
1	シークレット	青
2	ブリンク	赤
3	シークレット	紫（マゼンダ）
4	リバース	緑
5	リバースシークレット	水色（シアン）
6	リバースブリンク	黄
7	リバースシークレット	白

A XOR (full) 反転

COLOR〔＜ファンクションコード＞〕〔，＜ヌルキャラクターコード＞〕〔，＜グラフィックスイッチ＞〕

画面に対する色や機能を指定する。

ヌルキャラクターコード‥‥‥‥画面をクリアしたときに表示されるコード。アスキーコードか文字で指定する。通常は0。

グラフィックスイッチ‥‥‥‥‥1　グラフィックモード
0　キャラクターモード

CONSOLE〔＜スクロール開始行＞〕〔，＜スクロールの長さ＞〕〔，＜ファンクションキーの表示＞〕〔，＜カラーか白黒か＞〕

スクロールとは画面をせりあがらせること。指定した範囲がスクロールの対象になる。対象外の画面はそのままでせりあがらない。

ファンクションキー‥‥‥‥‥‥1　画面下のファンクションキーを表示。
0　画面下のファンクションキーを表示しない。

カラー指定‥‥‥‥‥‥‥‥‥‥1　カラーモード
0　白黒モード

GET@〔A〕（X1，Y1）－（X2，Y2），＜配列名＞〔，G〕

画面上の情報を指定した配列に読み込む。PUT@と一対で使われる。PUT@を参照のこと。

GET@ならPUT@で受ける。
GET@A（アトリビュートつき。色やリバースなどを同時にGETする）ならPUT@Aで受ける。
GET@‥‥‥‥G（グラフィックモード）ならPUT@‥‥‥‥＜条件＞で受ける。

LINE＜画面の行＞，＜ファンクションコード＞

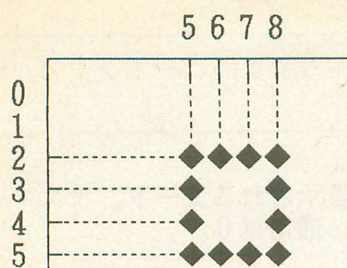
カラーモードの場合1行単位で，ファンクションを指定。
色はCOLORで指定する。

ファンクションコード‥‥‥‥0　ノーマル
1　ブリンク
2　リバース
3　リバースブリンク

白黒モードでは使えない。Syntax Errorになる。

LINE（X1，Y1）－（X2，Y2），"＜文字列＞"〔，＜ファンクションコード＞〕〔，B〔F〕〕

画面に文字を使って，線や箱を描く。色指定もできる。
BをつけるとB箱が描け，BFとすると箱のなかもうまる。



LINE (5, 2) − (8, 5), "◆", 4, B
など

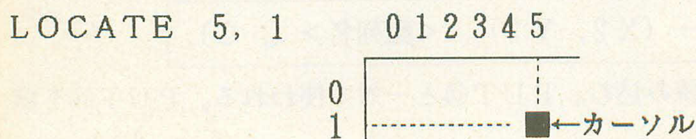
LINE (X1, Y1) − (X2, Y2), <P〔RE〕SET>〔, <ファンクションコード>〕〔, B〔F〕〕

画面にグラフィックモード（ドット）で線や箱を描く。
PRESETで消す。

LOCATE<水平位置>, <垂直位置>〔, <カーソルスイッチ>〕

画面上の指定位置へカーソルを移動する。
スイッチを0にするとカーソルは消滅する。

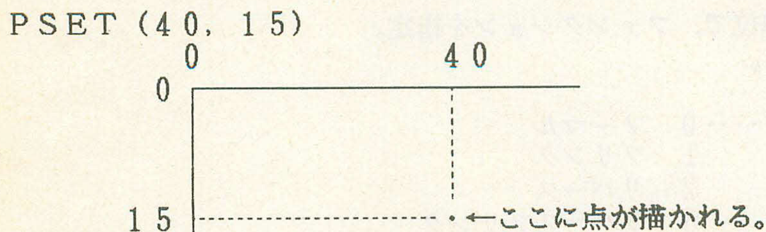
LOCATE 5, 1



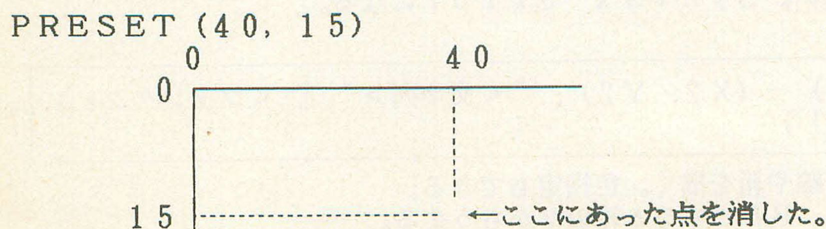
P〔RE〕SET (<水平位置>, <垂直位置>〔, <ファンクションコード>〕)

PSETは指定した位置に点（ドット）を描く。PRESETで消す。
水平位置…………………… 0から縦の桁数（けたすう）×2−1まで。
垂直位置…………………… 0から横の行数　　　　　×4−1まで。

PSET (40, 15)

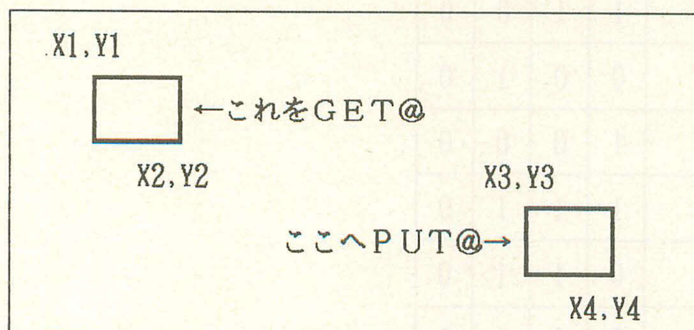


PRESET (40, 15)



PUT@〔A〕（X３，Y３）－（X４，Y４），<配列名>〔，<条件>〕

ＧＥＴ＠で作成した配列を画面の任意の部分に出力する。



ＧＥＴ＠とＰＵＴ＠の面積が同じなら，形がかわっても移せる。

```
10  LOCATE  10, 5:PRINT"ABCDE"
20  LOCATE   0, 0:DIM  P%(2)
30  GET@(10, 5)-(14, 5), P%
40  PUT@(20, 5)-(20, 9), P%
```

	10	11	12	13	14	15	16	17	18	19	20
5	A	B	C	D	E						A
6	上の５つの文字が ――――→										B
7										ここへ複写される	C
8											D
9											E

（キャラクターモード）ＧＥＴ＠ならＰＵＴ＠で受ける。

（Ａつき）ＧＥＴ＠ＡならＰＵＴ＠Ａ
アトリビュート（属性）つき。色やリバースなども同時に移せる。

（グラフィックモード）ＧＥＴ＠‥‥ＧならＰＵＴ＠‥‥<条件>
ＧつきＧＥＴ＠は，画面にドットがあれば１，なければ０と記憶していく。ＰＵＴ＠文の末尾の条件で次表のようになる。１はドットがある。０はない。

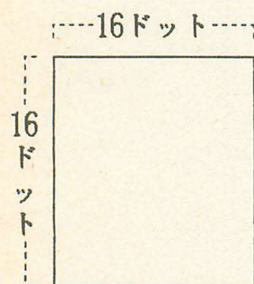
	GET@のドット	1		0	
	PUT@する画面の状態	1	0	1	0
条件	PSET	1	1	0	0
	PRESET	0	0	1	0
	AND	1	0	0	0
	OR	1	1	1	0
	XOR	0	1	1	0
	NOT	0	0	1	1

```
GET@（　0,　 0）－（15, 15）, A%, G
PUT@（25, 40）－（40, 55）, A%, OR
```

配列の大きさの計算のしかた

それぞれ割った数を切り上げて使う。
(1)キャラクターモード‥‥1文字しまいこむには1バイト（8ビット）必要。
　　必要なバイト数／b
(2)Aつきのとき
　　必要なバイト数×2／b
(3)グラフィックモード
　　（（必要なドット数）／8＋2）／b

bは ─┬ 整数型配　列（例P%）のとき2
　　　├ 単精度型配列（例A）　のとき4
　　　└ 倍精度型配列（例A#）のとき8



左のグラフィックモードを整数型配列P%（I）でとるには,
I＝（16*16／8＋2）／2
したがってP%（16）。配列はP%（0）からある。したがってP%（16）で17ある。

WIDTH〔<水平の桁数>〕〔, <垂直の行数>〕

画面に表示する文字の桁数, 行数を指定する。組み合わせは下のとおり。
　桁数　36, 40, 72, 80
　行数　20, 25

(5)　特殊ステートメント

BEEP〔<1または0>〕

内蔵ブザーから音を出す。

BEEP…………一定時間（約0．4秒）鳴ってとまる。
BEEP1………連続して鳴る。BEEP0（ゼロ）で停止する。

ERROR〔<エラーコード>〕

(1)エラー発生をシミュレートする。ON　ERROR　GOTOと組みにして使う。
(2)BASICで使っていないエラーコードを定義して，エラー処理ルーチンで使う。
　コードは0～255だが，BASICでエラーコードとして定義されているものと重複しないように注意する。

```
30  IF  A>1000  THEN  ERROR200………定義する。
80  IF  ERROR=200  THEN  ～  ……………使う。
```

KEY<キー番号>，"<文字列>"

f・1からf・10のファンクションキーを定義する。最大15文字まで文字列とコントロール文字を指定できる。

```
KEY1，"WIDTH40，20"+CHR$(13)
KEY2，"REM*****"
```

MOTOR〔<スイッチ>〕

カセットテープレコーダーのモーターを制御する。
スイッチ…………0…………モーターがとまる（off）
　　　　　　　　0以外の数….モーターが動く（on）
スイッチの指定がなければonをoffに，offをonにする。

ON　ERROR　GOTO<行番号>

エラー発生時のとび先を定義する。

TROFF

トレースモードを解除する。

TRON

トレースモードに入り，プログラムの実行状態を追跡する。実行したプログラムの行番号が次つぎにカギカッコで囲まれて画面表示される。

```
10 TRON
20 FOR K=5 TO 7
30 PRINT K
40 NEXT K
RUN
〔20〕〔30〕5………………ループ1回目，5がPRINTされる。
〔40〕〔30〕6………………ループ2回目，6がPRINTされる。
〔40〕〔30〕7………………ループ3回目，7がPRINTされる。
〔40〕
```

10行のTRONが実行されると，以後はトレースモード。TROFFで切る。

RESUME〔<0またはNEXTまたは行番号>〕

エラーに対する処理をすませたあと，RESUMEでプログラムの実行を再開する。

```
RESUME 0………………エラー行からプログラム再開する。
RESUME NEXT………エラー行の次の行から再開する。
RESUME <行番号>……指定の行から再開する。
```

ON ERROR GOTO <行番号>で，エラー処理ルーチンへとぶ。エラー処理したら，プログラムを終了させるのか再実行させるのかの判断が必要になる。したがってエラー処理ルーチンの最後には，ENDかRESUMEが要る。

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

(例) WAIT

キーボードの入力ポートは，キーが押されないかぎり255（&HFF）の状態である。すなわち11111111とビットがすべて立つ。WAITも入力ポートのデータを読むが，データがある状態になるまで待つ。キーボードマップ参照のこと。

WAIT P，J，K　　P‥‥ポート番号。　J‥‥データ
　　　　　　　　K‥‥セットデータ。セットしなければ0とみなす。

まず，P XOR K を実行する。結果をQとすれば，Q AND J を実行して，0であれば再度ポートの状態を読んで同じことを繰り返す。0でなければ次のプログラムへ進む。

たとえば，テンキーの「3」が押されたら，ABCDEFGとプリントするプログラムは次のようになる。

```
10 WAIT 0, 8, 255
20 PRINT "ABCDEFG"
```

```
なにもしないPの状態＝255＝1 1 1 1 1 1 1 1
                 K＝255＝1 1 1 1 1 1 1 1
          ――――――――――――――――――――――――――――――
   (P XOR K)＝ Q  ＝0 0 0 0 0 0 0 0
             J＝ 8  ＝0 0 0 0 1 0 0 0
          ――――――――――――――――――――――――――――――
        (Q AND J)＝0 0 0 0 0 0 0 0

3のキーを押したPの状態＝247＝1 1 1 1 0 1 1 1
                 K＝255＝1 1 1 1 1 1 1 1
          ――――――――――――――――――――――――――――――
   (P XOR K)＝ Q  ＝0 0 0 0 1 0 0 0
             J＝ 8  ＝0 0 0 0 1 0 0 0
          ――――――――――――――――――――――――――――――
        (Q AND J)＝0 0 0 0 1 0 0 0＝8
```

結果が0でないから，次の20行が実行されてABCDEFがプリントされる。
WAITは使いかたによっては無限ループに入ってしまう。30 GOTO 10 を追加すると，STOPキーではとまらない。STOPキーを押しながらリセットキーを押せばプログラムは助かる。

一般的な使いかたは，たとえばプリンターがBUSY（稼働中）かREADY（待機中）かをWAITを使ってステータスビットを調べて判断し，OUTでデータを送りだす。さもないと，ほかの仕事で稼働中のところへさらにOUTでデータを送ってしまい，データを消滅させてしまう。

(例) INP

Kのキーをおすとベルがなる。

```
10 IF INP (3) =247 THEN 30
20 GOTO 10
30 BEEP1 :FOR J =0 TO 50:NEXT :BEEP0 :GOTO 10
```

INP（x）は，その他の関数にある通りポート番号x番から入力したデータを与える。わかりやすく，キーボードで説明してみる。
キーボードマップをみると，縦に00から09までのインプットアドレスがあって，横にD0からD7までのデータバスがある。コンピューターが稼働中なら，常時00から09のインプットアドレスを監視して，キーが押されたかどうか見張っている。キーがなにも押されない状態では，D0からD7はすべてビットが立っている（負論理という。255すなわち&HFFの状態）。キーが押されるとそこだけ1が0になる。

	1	2	4	8	16	32	64	128
	D0	D1	D2	D3	D4	D5	D6	D7←D7が最上位ビット
キーが押されていない	1	1	1	1	1	1	1	1＝255＝&HFF
「1」が押された	1	0	1	1	1	1	1	1＝253＝&HBF
「K」が押された	1	1	1	0	1	1	1	1＝247＝&HEF

数値関数

xは数値を示す

関　　数	機　　能	備　　考
ABS（x）	xの絶対値｜x｜を与える	PRINT　ABS（－40）は40 PRINT　ABS（A＋8）など
ATN（x）	xのアークタンジェント	xはラジアン。ラジアンと変数の関係は1ラジアン＝180°／π ∴1°＝3．14159／180
CDBL（x）	xを倍精度数値に変換	A＝CDBL（x）など
CINT（x）	xを整数に変換	PRINT　CINT（34．56）は34 －32768≦x≦32767のこと
COS（x）	xのコサイン	xはラジアン 1ラジアン＝180°／π
CSNG（x）	xを単精度変数に変換	A＝CSNG（x＋2）など
EXP（x）	eのx乗	PRINT EXP（1．60944）は5．00001 e＝2．71828459 LOGとの関係はy＝logex x＝e＾y
FIX（x）	小数点以下を切りすてた整数	PRINT　FIX（4．56）は4 PRINT　FIX（－4．5）は－4
INT（x）	xを超えない最大の整数	PRINT　INT（4．56）は4 PRINT　INT（－4．56）は―5
LOG（x）	xの自然対数	x＞0のこと。自然対数を常用対数にする式はLOG（x）／LOG（10）

数値関数

xは数値を示す

関　　　数	機　　　能	備　　　考
RND (x)	乱数をつくる	xはダミー。結果は0<乱数<1になる。A=RND (1) など x>0なら通常の乱数 x=0なら同系列乱数 x<0なら乱数系列を更新する
SGN (x)	xの正負符号を与える	A=SGN (x) x>0　なら　A=1 x=0　なら　A=0 x<0　なら　A=−1
SIN (x)	xのサイン	xはラジアン 1ラジアン=180°/π
SQR (x)	xの平方根	A=SQR (2) など。x≧0であること
TAN (x)	xのタンジェント	xはラジアン 1ラジアン=180°/π

ストリング関数

x, yは数値, Sは文字列とする

関数	機能	備考
ASC (S)	文字列Sの最初のコードを数値で与える	A$="ABC" PRINT ASC (A$) は65 BとCは無視される
CHR$ (x)	コードxの対応する文字や機能を与える ASCの逆	PRINT CHR$ (7) はBEEP1 PRINT CHR$ (84) はT PRINTCHR$ (&HF1) は円
HEX$ (x)	数値xの16進数文字列を与える	PRINT HEX$ (80) は50
INKEY$	キーボードが押されたかどうか調べる。キーボードが押されればその文字を与える	10 X$=INKEY$ 20 IF X$="" THEN 10
(ディスク) INPUT$ (x〔, 〔#〕y〕)	キーボードまたは番号yのファイルから入力したx文字分の文字列	キーボードから入力された文字は画面表示されない。すべてのコントロール文字はコントロールCをのぞき, そのまま入力される X$=INPUT$ (2, #1) など
INSTR (〔x, 〕S1, S2)	S1からS2と同じ文字列を探し, 合致したら最初の文字がS1の何文字目かを与える 同じ文字列がみつからなければ0	X$="ABCABCD" : Y$="CD" INSTR (X$, "C") は3 INSTR (4, X$, Y$) は6 xを与えるとS1のx文字目から探しはじめる
LEFT$ (S, x)	Sの左からx番目までの文字列を与える	X$="ABCDEF" LEFT$ (X$, 3) はABC
LEN (S)	Sが何文字あるかを与える	X$="ABCDEF" PRINT LEN (X$) は6
MID$ (S, x, y)	Sの中の左からx文字目から長さyの文字列を与える 次ページへ続く	X$="ABCDEF" Y$=MID$ (X$, 3, 2) : PRINT Y$はCD

ストリング関数

x，yは数値，Sは文字列とする

関　　　数	機　　　能	備　　　考
MID$ つづき	MID$は式の左辺でも使うことができる	10 A$＝"AAAAA" 20 MID$（A$，2，2） ＝"●●" 30 PRINT A$ 40 END 上例では20行でMID$が式の左辺に使われている プログラムを実行すると，結果は次のようになる A●●AA
OCT$（x）	数値xの8進数文字列を与える	PRINTOCT$（&H11）は21
RIGHT$（S，x）	Sの右からx番目までの文字列を与える	X$＝"ABCDEF" PRINTRIGHT$（X$，3）はDEF
SPACE$（x）	x個の空白を与える	X$＝"ABC"：Y$＝"DEF" C$＝X$＋SPACE$（7） ＋Y$ PRINT C$は ABC←スペース7個→DEF
STR$（x）	数値xをそれを表示する文字列に変換	PRINT STR$（1234）＋"円"は 1234円
STRING$（x，<Sまたはy>）	Sの最初の文字をx個または，アスキーコードyの文字をx個与える	PRINT STRING$（5，"ABC"）は AAAAA PRINT STRING$（5，&HEC）は ●●●●●
VAL（S）	数字を数値へ変換する	Sの最初の文字が＋，－，&，数字でなければ0になる PRINT VAL（"010 "）は10 PRINT VAL（"A10 "）は0

ディスク関数

ｘは数値，Ｓは文字列とする

関　　数	機　　能	備　　考
ＣＶＤ（Ｓ）	８バイトの文字列を倍精度の数値へ変換する	ランダムファイルから読み出した数値は，文字列から数値に変換する必要がある １０　ＦＩＥＬＤ＃１，４ＡＳ　Ａ＄，２０ＡＳ　Ｂ＄‥‥ ２０　ＧＥＴ＃１ ３０　Ａ＝ＣＶＳ（Ｓ）
ＣＶＳ（Ｓ）	４バイトの文字列を単精度の数値へ変換する	
ＣＶＩ（Ｓ）	２バイトの文字列を整数の数値へ変換する	
ＤＳＫＩ＄（＜ドライブ番号＞，＜トラック番号＞，＜セクター番号＞）	指定したセクターの内容を変数名と＃０のフィールドバッファに与える	Ａ＄＝ＤＳＫＩ＄（１，３，５） ＤＳＫＯ＄ステートメントと逆の働きをする
ＥＯＦ（＜ファイル番号＞）	指定した番号のシーケンシャルファイルの終わりを示す	終りになると－１（真）の値をとる。シーケンシャルファイルからの入力時，”ＩＮＰＵＴ　ＰＡＳＴ　ＥＮＤ”エラーをふせぐために使う
ＦＰＯＳ（＜ファイル番号＞）	ファイルのあるセクター番号	セクター番号が与えられることを除くとＬＯＣと同じ
ＬＯＣ（＜ファイル番号＞）	ランダムファイルの場合レコード番号を指定しないＧＥＴまたはＰＵＴが実行されたときに，使用されたレコード番号が与えられる。シーケンシャルファイルの場合，ファイルが開かれてから読み出された（または書き込まれた）セクター数が与えられる	２０　ＰＲＩＮＴ　ＬＯＣ（１）

ディスク関数

xは数値，Sは文字列とする

関数	機能	備考
LOF（＜ファイル番号＞）	ランダムファイルのPUT，GETによりアクセスされた最大のレコード番号を与えられる	500IF NO＞LOF（1） THEN PRINT ”ファイル END”
MKD$（x）	倍精度の数値xを8バイトの文字列へ変換する	LSET，またはRSETによりランダムバッファに入れる数値はすべて文字列に変換しなければならない 10 OPEN”FILE”AS #1 20 FIELD#1，20AS N$，4 AS A$ 30 X$=”ABC”：LSET N$=X$ 40 A=34.56 50 RSET A$=MKS$（A）
MKS$（x）	単精度の数値xを4バイトの文字列へ変換する	
MKI$（x）	整数xを2バイトの文字列に変換する	
ATTR$（＜ドライブ番号＞または＜ファイル番号＞または＜ファイル名＞）	指定したドライブやファイルの現在の属性を示す	属性を返す。 R‥‥READ AFTER WRITE P‥‥WRITE PROTECT A$=ATTR$（1）
DSKF（＜ドライブ番号＞）	指定したドライブにマウントされているディスクの未使用クラスター数を与える	PRINT DSKF（1）

その他の関数

xは数値，Sは文字列とする

関　　数	機　　能	備　　考
CSRLIN	画面上のカーソルの縦の位置を与える	LOCATE 4，5：PRINT CSRLIN は 5と出る POS参照のこと
DATE＄	内蔵クロックの日付	PRINT DATE＄ 日付をセットするには， DATE＄＝"YY／MM／DD" で入る DATE＄＝"81／04／12" などと入れる
ERL	エラーが起きた行番号を与える	IF ERL＝130 THEN 50
ERR	エラーの番号を与える	IF ERR＝4 THEN 100
FRE（x）	ユーザー用メモリーエリアの空きを示す。xはダミー	PRINT FRE（0） 電源投入後，RAM16Kのときは，10402，32Kならば26786
INP（x）	I／Oアドレスxのポートから読んだデータの値	A＝INP（2） 33ページに例がある。キーボードマップとあわせて参照のこと
LPOS（x）	プリンターのヘッドの位置。xはダミー	1行60文字で改行したければ，最大80文字の場合 IF 80－LPOS（1）＝20 THEN LPRINT
PEEK（x）	メモリーx番地の内容を与える	A＝PEEK（&H1056） 出力ステートメントPOKEの逆
POINT（<水平位置>，<垂直位置>）	指定した位置にドットがあるかどうかを判断する	あれば－1（真），なければ0（偽）となる 50 IF POINT（30，40）＝0 THEN RETURN

その他の関数

ｘは数値，Ｓは文字列とする

関　　　数	機　　　能	備　　　考
ＰＯＳ（ｘ）	画面上のカーソルの横の位置を与える	ＰＲＩＮＴ　ＰＯＳ（０），ＣＳＲＬＩＮ これでカーソルの水平，垂直位置がわかる
ＳＰＣ（ｘ）	プリント文中で使い，ｘ個の空白を与える	ＰＲＩＮＴ　”Ａ”；ＳＰＣ（３）；”Ｂ” Ａ△△△Ｂ （△は空白とする）
ＴＡＢ（ｘ）	プリント文中で使い，左端からｘ字まで空白とする。現在のカーソル位置＞ｘならば無効	ＰＲＩＮＴ　ＴＡＢ（５）；”Ａ” △△△△△Ａ
ＴＩＭＥ＄	内蔵クロックの時刻	時刻をセットするには ＴＩＭＥ＄＝”ＨＨ：ＭＭ：ＳＳ” ＴＩＭＥ＄＝”２３：００：１５”
ＵＳＲ<ｎ>（ｘ）	ユーザーが定義したｎ番目の機械語ルーチンへとぶ	ｎは０から９まで。省略すると０ ｘは引数 ステートメントＤＥＦＵＳＲ参照
ＶＡＲＰＴＲ（<変数名>） ＶＡＲＰＴＲ（＃<ファイル番号>）	変数の割り当てられているアドレスを与える。ファイル番号を指定した場合には，指定したファイル番号のディスクファイルに与えられた入出力バッファの開始アドレスを与える	アドレスは３２７６７から－３２７６８まで。負の場合には，６５５３６を加えると実数値になる

PRINT USING 一覧表

左づめで表示	!	与えられた文字列の最初の1文字だけをプリントする 10 A$="ABC":B$="DEF" 20 PRINT USING "!";A$;B$ AD
	& &	文字データの桁数を指定。空白+2が桁数 10 A$="ABC":B$="DEF" 20 PRINT USING "& &";A$ ABC
右づめで表示	#	数字の桁数を指定 10 A=123 20 PRINT USING "####";A 123 30 PRINT USING "タイジ゛ュウ###.#Kg";56.7 タイジ゛ュウ 56.7Kg
	.	#と#の間へ.(ピリオド)をうつと，そこが小数点 10 PRINT USING "####.###";A 123.456 20 PRINT USING "####.##";A 123.45
	+	正の符号をつける。数値が負のときは負の符号がつく PRINT USING "+##.##";77.8 +77.8 PRINT USING "##.##+";-12.34 12.34-
	-	負の符号をつける。数値が正のときは正の符号はつかない PRINT USING "##.##-";-68.95 68.95- PRINT USING "##.##";-5.67 -5.67
	**	頭部の余白を*でうめる PRINT USING "**###.##";12.3 ***12.30 PRINT USING "**###.#円";2.54 ****2.5円
	¥¥	数値の頭に¥をつける PRINT USING "¥¥#####";2000 ¥2000

23 43 63

PRINT USING 一覧表

右づめで表示	**¥	前の2つの合成 PRINT USING "**¥#####";1234 ***¥1234
	,	整数部分を3桁ごとに，（コンマ）で区切る PRINT USING "##########,";123456789 123,456,789
	^^^^	指数型表示にする 10 A=123 20 PRINT USING "##.##^^^^";A 1.23E+02 30 PRINT USING "##.##^^^^";-0.000005 -5.00E-06
%	数字の桁（#）が足りないと頭に%が表示される 数値を丸めた結果桁上りして桁（#）が不足すると%が表示される PRINT USING ".##";0.993 %1.00	

LPRINT USING とするとプリンターへ出力する。

<例>

```
10 A$="ASAHISHIMBUN"
20 B$="KS"
30 C=1000
40 LPRINT USING "&←空白10個→&  && ####円";
               A$,B$,C
```

結果

```
ASAHISHIMBUN KS 1000円
```

浮動小数点形式による数の表示

N－BASICでは，実数（1．5や3．14159など）はすべて浮動小数点形式で扱っている。
これは，365なら次のような表現になる。

$$3.65 \times 10^{2}$$

3．65を仮数，2を指数と呼ぶ。この方法を2進数でやると次のようになる。

10進	2進
0．5	1.0×2^{-1}
1．0	1.0×2^{0}
1．25	1.01×2^{0}
1．5	1.1×2^{0}
2．0	1.0×2^{1}
0．0	0.0×2^{0}

N－BASICの内部表現では，次のようになる。

＊単精度　　4バイトで表現する

下位8ビット	中位8ビット	上位8ビット	指数8ビット
仮数			

＊倍精度　　8バイトで表現する

最下位8ビット	…………	最上位8ビット	指数8ビット
仮数			

＜指数＞
指数は正負を表すため，実際の数値に129（&H81）を足した値を使っている。

指数の値	＋126	＋1	0	－1	－128
内部表現	255	130	129	128	1

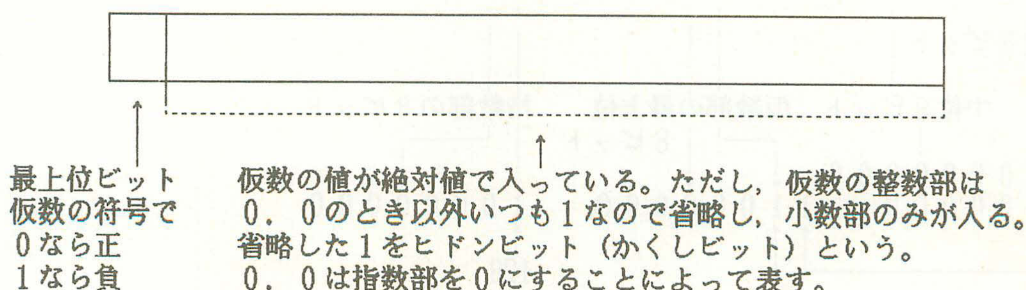
$$1 = 1.0 \times 2^{129-129} = 1.0 \times 2^{0} = 1.0 \times 1$$

$$2 = 1.0 \times 2^{130-129} = 1.0 \times 2^{1} = 1.0 \times 2$$

$$0.5 = 1.0 \times 2^{128-129} = 1.0 \times 2^{-1} = 1.0 \times 0.5$$

＜仮数＞

仮数は，正規化（桁あわせ）という操作によって上位ビットにつめられている。
仮数部を上位バイトから下位バイトへとならべると下図のようになる。



２進数で表した仮数	仮数部の内部表現
１．０００------	０　０００------
１．０１０------	０　０１０------
１．１００------	０　１００------
－１．０００------	１　０００------

（左：↑仮数の整数部　仮数の小数部）（右：↑仮数部の符号　仮数の小数部）

以上の方法で　－０．７５　を内部表現になおすと次のようになる。

$$-0.75 = -0.11 \times 2^{0} = -1.1 \times 2^{-1} = -1.1 \times 2^{128-129}$$

（-0.11×2^{0} → -1.1×2^{-1}：正規化）

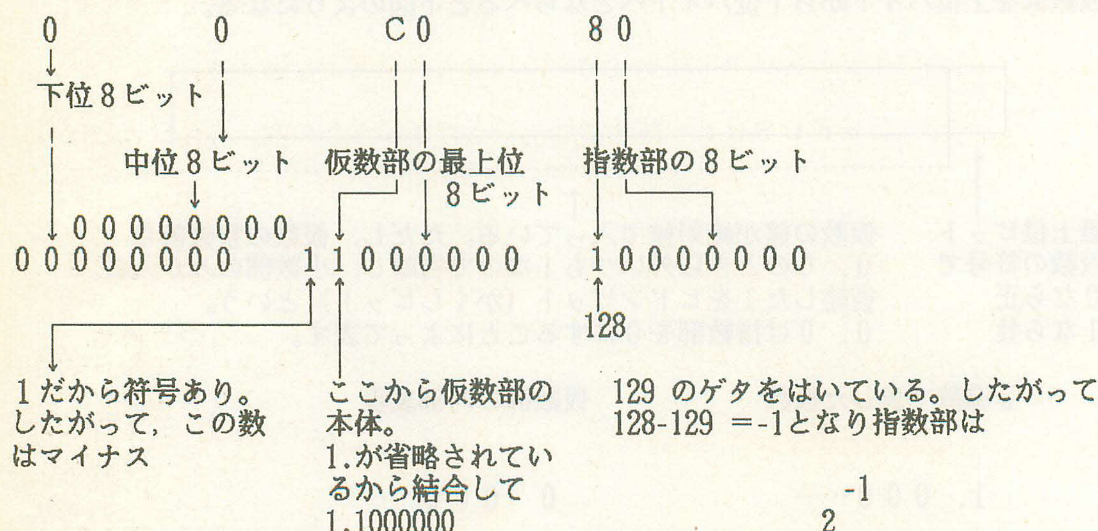
＊内部表現を画面に表示する方法

```
100  A=－0．75
200  FOR  J=VARPTR（A） TO  VARPTR（A）＋n
300  PRINT  HEX$（PEEK（J））；”  ”；
400  NEXT  J
```

（n：単精度なら3、倍精度なら7）

VARPTRは変数Aがしまわれている最初の番地を返すので，その番地から必要バイト数をPEEKで取り出して表示する。

実行結果は次のようになる。



以上をまとめると次のようになる。

$$-1.10000000000000000000000 \times 2^{-1} = -0.11$$

2進法の掛算だから桁がひとつずれて －0．11になる

2^{-2} の位＝0．25
2^{-1} の位＝0．5
→ 0．75

（参考）

$2^{-1} = 1/2 = 0.5$　　　$2^{-2} = 1/4 = 0.25$

$2^{-3} = 1/8 = 0.125$　　　$2^{-4} = 1/16 = 0.0625$

ＵＳＲ関数

Ｎ－ＢＡＳＩＣでは，機械語サブルーチンをＵＳＲ関数（ユーザー関数）として使うことができる。機械語サブルーチンを使う手順は次のようにする。

（1）　ＣＬＥＡＲ文でサブルーチンを入れる場所を確保する。
通常ＢＡＳＩＣで使うアドレスの上限は５９９０３（＆ＨＥ９ＦＦ）番地だから，ここから機械語で必要とするスペース分だけ引いてセットしてやればよい。２００語分必要なら次のようになる。（ＣＬＥＡＲ文参照）

ＣＬＥＡＲ　３００，５９９０３－２００

（2）　確保した場所に，ＰＯＫＥ文やモニター（ＭＯＮ）で機械語を書きこむ。

（3）　機械語サブルーチンの開始番地をＤＥＦＵＳＲ文で定義する。

ＤＥＦＵＳＲｎ＝開始番地　（ｎは0から９の数，与えなければ0）

（4）　機械語サブルーチンを呼び出して使う。

Ｘ＝ＵＳＲｎ（引き数）　（ｎはＤＥＦＵＳＲのｎと一致させる）

引き数の型は，整数型，単精度実数型，倍精度実数型，文字型とある。
引き数が渡されると，Ａレジスターには引き数の型（ＦＡＣすなわち浮動小数点アキュムレーターの必要バイト数）が入る。ＨＬペアレジスターには，引き数が格納されているＦＡＣへのポインターの番地が入る。引き数が文字の場合には，ＤＥペアレジスターがストリングディスクリプターの番地を指す。

引き数の型		整数型	単精度実数型	倍精度実数型
Ａレジの値		2	4	8
浮動小数点アキュムレーター	FAC7			仮数の最下位８ビット
	FAC6			仮数の第６位８ビット
	FAC5			仮数の第５位８ビット
	FAC4			仮数の第４位８ビット
HLレジスターはここを指す→	FAC3	引き数の下位８ビット	仮数の下位８ビット	仮数の第３位８ビット
	FAC2	引き数の上位８ビット	仮数の中位８ビット	仮数の第２位８ビット
	FAC1		仮数の上位８ビット	仮数の最上位８ビット
	FAC0		指数部	指数部

ＨＬレジスターは常にＦＡＣ３の番地を指す。実数型は浮動小数点形式になっている。また，ストリングディスクリプターは次のようになっている。

<table>
<tr><td>引き数の型</td><td colspan="2">文　字　型</td></tr>
<tr><td>Ａレジの値</td><td colspan="2">３</td></tr>
<tr><td rowspan="3">DEレジスター→
ーはここの
番地をさす</td><td>文字の長さ</td><td>８ビット</td></tr>
<tr><td colspan="2">実際に文字が格納されている番地の下位８ビット</td></tr>
<tr><td colspan="2">実際に文字が格納されている番地の上位８ビット</td></tr>
</table>

（例）任意の整数に５を足す（整数型引き数の例）

```
 10  INPUT A%
 20  B%=A%+5
 30  PRINT  B%
```

上のプログラムの２０行を機械語サブルーチンにしてみる。

```
 10  CLEAR 300, &HDFFF
 20  DEFUSR4=&HE000
 30  FOR J=0 TO 9
 40  READ M:POKE &HE000+J, M
 50  NEXT J
 60  DATA &H7E, &HC6, &H05, &H77, &H23
 70  DATA &H7E, &HCE, &H00, &H77, &HC9
 80  INPUT  A%
 90  B%=USR4 (A%)
100  PRINT  B%
```

３０行～７０行は，モニターモード（MON）で＊ＳＥ０００として入力してもよい。初心者のために機械語の説明をつけておく。

```
7E→LD  A, (HL)      HLレジスターの指す番地の内容（A%の下位8ビッ
                     トが入っている）をAレジスターへ入れる
C605→ADD  A, 5      Aレジスターに5を加える
77→LD (HL), A       Aの値をHLレジスターの指す番地へもどす
23→INC  HL          HLレジスターを+1する
7E→LD  A, (HL)      +1されたHLレジスターの指す番地の内容（A%の
                     上位8ビットが入っている）をAレジスターへ入れる
CE00→ADC  A, 0      下位ビットの演算結果のキャリー部分を加えてやる
77→LD (HL), A       Aの値をHLレジスターの指す番地へもどす
C9→RET              サブルーチンの終わり
```

詳細は，機械語→ニーモニック変換表をみよ。

（例）　キーボードから入力した文字を画面表示する（文字型引き数の場合）
N－BASICの画面表示ルーチン（&H0257～&H02CA番地）を使う。すなわち，サブルーチンの中からさらにサブルーチンへ飛ぶ。

```
10 CLEAR 300，&HDFFF
20 DEFUSR3＝&HE000
30 INPUT A$
40 X$＝USR3（A$）
```

機械語の部分は，モニターモード（MON）にして＊SE000から，13 1A 6F 13 1A 67 7E CD 57 02 C9と入力して，すんだらCTRLキーと「B」を押してBASICモードへもどる。

```
13→INC DE              DEを1番地進める。その番地に文字が格納
                        されている番地の下位8ビットがある
1A→LD A （DE）         Aレジスターへ
6F→LD L，A             Lレジスターへ取り出す
13→INC DE              さらに1番地進める
1A→LD A （DE）         上位8ビットをAレジスターへ
67→LD H，A             Hレジスターへ取り出す
7E→LD A，（HL）        1文字目をAレジスターへ取り出す
CD5702→CALL 0257       画面出力ルーチンへ
C9→RET                 サブルーチンの終わり
```

（例）　画面をプリントする　→ COPY
PC－8001は，ターミナルモード時の画面プリント機能（&H124A番地）を持っている。もともとある機能だから，CLEARで場所を確保しなくてもよい。

```
DEFUSR2＝&H124A   画面プリントサブルーチンの入口アドレスを定
                  義する。
X＝USR2（0）      画面プリントサブルーチンを実行する。
```

使用上の注意点
＊この機能は，文字しかできない。ドットグラフィックは，空白となる。
＊画面にカーソル移動で表示したものは，そのままでは左に詰まってプリントされてしまう。これを避けるには，COLOR文のヌルキャラクターコードで0（ヌル）ではなく，32（スペース）を指定し，画面にスペースを埋めておく。

ポイント
ファンクション・キーに　X＝USR2（0）を定義しておくと便利。ダイレクトモードで次のコマンドを実行すれば，画面プリントができる。

```
DEFUSR2＝&H124A
KEY1，"X＝USR2（0）"＋CHR$（13）
```

以後 [f・1] を押せば，画面プリントされる。

＊ユーザールーチン使用上の注意点

① 引数と結果の型が違う場合

引数が整数でない場合でも，ＢＡＳＩＣ内のMAKＩNTと呼ばれるルーチンを経由してリターンすることにより，整数値を返すことができる。
返したい整数値はＨＬレジスターに入れておく。

ＰＵＳＨ	ＨＬ	ＨＬレジスターの内容を，スタックにいれる
ＬＤ	ＨＬ，（００２４）	ＨＬレジスターに，MAKＩNTルーチンのアドレスをいれる （００２４）に，MAKＩNTルーチンのアドレス２７９Ｃが入っている
ＥＸ	（ＳＰ），ＨＬ	スタックの内容と，ＨＬレジスターの内容とをいれかえる。これで，ＨＬレジスターには返す値が，スタックにはMAKＩNTルーチンのアドレスが入る
ＲＥＴ		MAKＩNTルーチンのアドレスをリターンアドレスとみなし，ジャンプする

② 文字列引数として文字定数を渡すとき

文字定数はプログラム内にあるため，これに対する操作はプログラム中の文字定数を変化させることになる。これを避けるためには，次のように式の型にすることにより，いったん文字領域へコピーしてから引数とする。Ａ＄＝”ＡＢＣＤ”＋””

③ ユーザールーチンのエラー

ユーザールーチンはＢＡＳＩＣとは全く別に動くため，エラーメッセージは表示されない。そればかりでなく，ＢＡＳＩＣで書かれたプログラムをこわしてしまうこともある。
当然，ＳＴＯＰもきかないため，リセットしか止める方法がない。そこで，プログラムの中に，デバッグ用として画面表示を呼び出す命令を入れておき，どこまでプログラムが動いたかを見ながらまとまった仕事単位にデバッグしていくとよいだろう。

④ 機械語サブルーチンでのスタック

ＢＡＳＩＣは，ユーザールーチン用スタックとして自動的に８レベル，１６バイトを確保し，スタックポインターをセットしてくれる。これより大きいスタックが必要な場合には，自分でスタックをつくらなくてはならない。
Ｅ０００Ｈ番地からＥ０ＦＦＨ番地までの２５６バイトを使うなら１２８レベルのスタックとなる。
機械語サブルーチンの始めでスタックポインターを退避し，スタックポインターにはＥ１００Ｈをロードする。これは，ＰＵＳＨ命令などのスタックを使う命令がスタックポインターの指すひとつ手前の番地からスタックを使うため。（ＰＵＳＨ命令の説明参照　１０４ページ）
機械語サブルーチンからＢＡＳＩＣに戻るには，機械語のＲＥＴ命令を実行する。ただし，ＲＥＴ命令で使われる戻り番地はもとのスタックにあるので，スタックポインターに始めに退避した値をロードしなければならない。

＊スタックの作り方

ニーモニックで書くと次のようになる

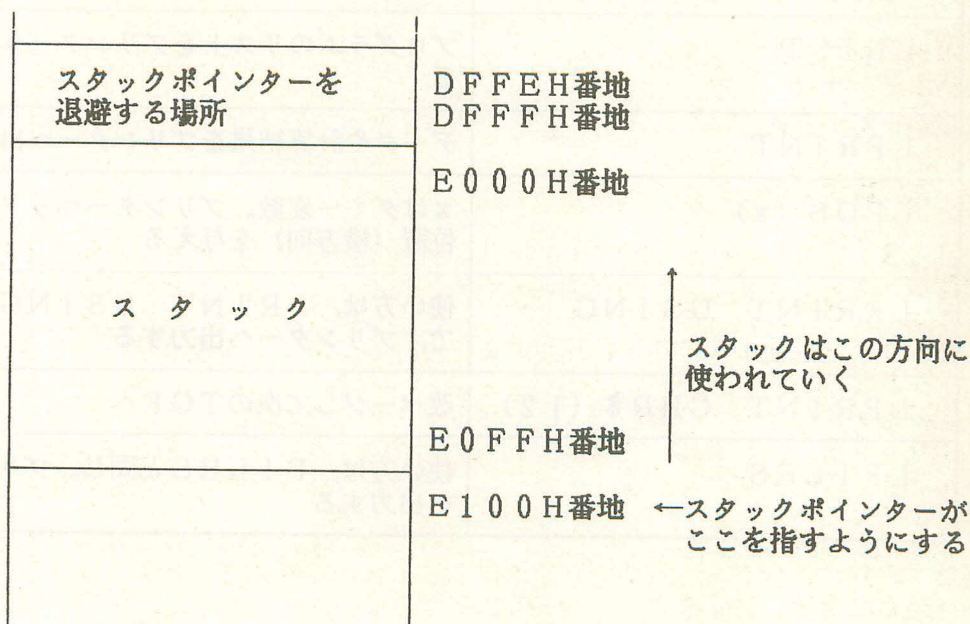
```
LD   (0DFFEH), SP     スタックポインターを退避する
LD   SP, 0E100H       スタックポインターを新しいスタックにセットする
      ・
      ・サブルーチン本体
      ・
LD   SP, (0DFFEH)     スタックポインターを復帰する
```

スタックをE000H番地からE0FFH番地につくる。
スタックポインターをDFFEH番地, DFFF番地に退避する。

機械語で書くと次のようになる

```
ED73FEDF        LD   (0DFFEH), SP
3100E1          LD   SP, 0E100H
      ・
      ・サブルーチン本体
      ・
ED7BFEDF        LD   SP, (0DFFEH)
```

(機械語では上位下位バイトが逆になるのでFEDF。104ページ参照)

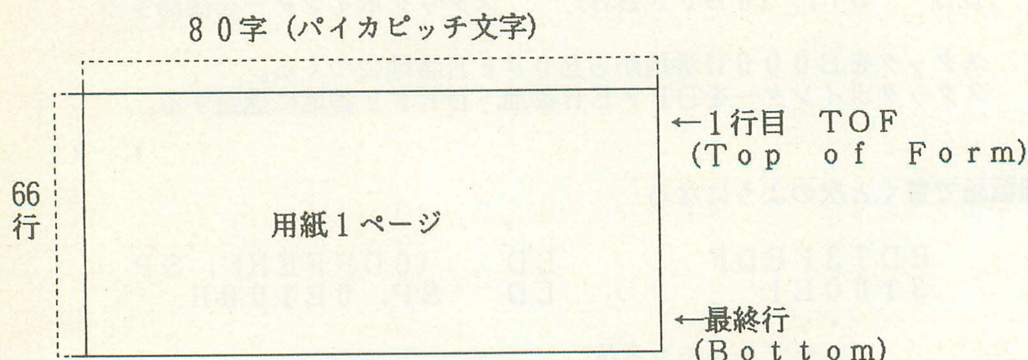


プリンター（PC－8023）の使い方

＜基本事項＞

（1）コンピューター本体とプリンターとが正しく接続されていること。
（2）電源が入っていること。
（3）オンライン状態であること（セレクト（SEL）スイッチが押されている）。
（4）用紙が正しくセットされていること。

＜印字文字数＞



＜基本操作＞

命　　令	機　　能
LLIST	プログラムのリストをプリンターへ出力する
LPRINT	データや計算結果をプリンターへ出力する
LPOS (x)	xはダミー変数。プリンターヘッドの現在位置（横方向）を与える
LPRINT USING	使い方は，PRINT USINGと同じで，プリンターへ出力する
LPRINT CHR$ (12)	改ページして次のTOFへ
LFILES	使い方は，FILESと同じ。プリンターへ出力する

<文字制御>　パワーオン時には，パイカで英数記号とカタカナに設定されている。縮小文字の強調印字はできない。

機　　能	命　　　　　令
パイカにする	LPRINT CHR$ (27) ;CHR$ (78) ;
エリートにする	LPRINT CHR$ (27) ;CHR$ (69) ;
縮小文字にする	LPRINT CHR$ (27) ;CHR$ (81) ;
プロポーショナル	LPRINT CHR$ (27) ;CHR$ (80) ;
英数記号とカタカナ	LPRINT CHR$ (27) ;"$";
英数記号とひらがな	LPRINT CHR$ (27) ;"&";
拡大文字にする	LPRINT CHR$ (14) ;
拡大文字の解除	LPRINT CHR$ (15) ;
強調文字にする	LPRINT CHR$ (27) ;"!";
強調文字の解除	LPRINT CHR$ (27) ;CHR$ (34) ;
アンダーラインを引く	LPRINT CHR$ (27) ;CHR$ (88) ;
アンダーラインの解除	LPRINT CHR$ (27) ;CHR$ (89) ;
ドットをプリントする	LPRINT CHR$ (27) ;CHR$ (83) ; "n"　　nは送るバイト数で4桁の10進数

ドットを使って漢字の「江」をプリントするには，まず16×16のマス目に文字パターンを作る。マス目の1つが1ドット。ぬりつぶされたマスがビットのON，空白がOFF と考え，縦8マス分を8桁の2進数とする。左から右へ16回やれば文字の上半分のDATAができる。SUBROUTINEでデータをPRINTER へ送り，上半分を印字する。改行して下半分を印字する。小漢字にしたければ，縦16マスを奇数マスと偶数マスで別々にDATAを作り，10行の改行幅を04とする。奇数マスが先にうたれ，2ドット改行，偶数マスがうたれる。

```
 10 LPRINT CHR$ (27) ;"T ";"16";
 20 FOR J =1 TO  2 :FOR K =1 TO 16
 30 READ A:LPRINT CHR$ (27) ;CHR $ (83) ;"0001"; :GOSUB 500
 40 NEXT K:LPRINT  :NEXT J:END
 50 DATA 0,64,130,4,8,0,0,4,4,4,4,252,4,4,4,4
 60 DATA  0,32,64,192,32,16,0,64,64,64,64,127,64,64,64,64
500 OUT &H10,A :OUT &H40,0 :OUT &H40,1
510 S =INP (&H40 ) AND 1 :IF S=0 THEN RETURN ELSE 510
```

小漢字のDATAは，
```
50 DATA 0,0,66,0,68,0,34,2,2,2,126,2,2,2,2,0
60 DATA 0,41,8,74,8,32,64,64,64,64,126,64,64,64,64,0
```

<幅と行と位置の制御>

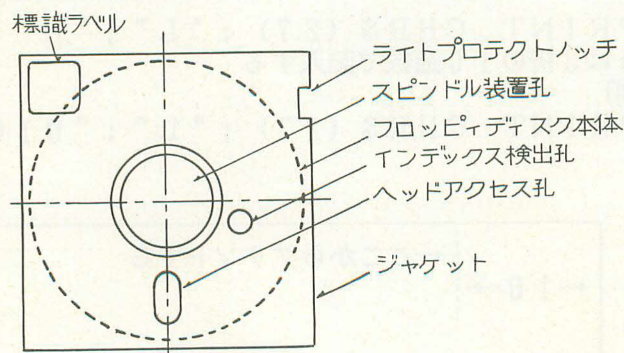
機　　能	命　　令
改行幅を1/6インチにする	LPRINT CHR$(27);"A";
改行幅を1/8インチにする	LPRINT CHR$(27);"B";
n/144インチの改行幅にする	LPRINT CHR$(27);"T";"n"; n=01～99, また2/144は1ドット幅となる
水平タブを設定する	LPRINT CHR$(27);CHR$(40);"a,b,c,……..p."; a～pは3桁の10進数。10は010と書くこと コンマ(,)は継続コード，ピリオド(.)は終了コード a～pは16個まで設定できる (例) 100 LPRINT CHR$(27);CHR$(40);"010,020,030."; 110 FOR I=1 TO 4 120 LPRINT I; 130 GOSUB500 140 NEXT I 150 END 実行結果 1　2　3　4 ←10→ ←20→ ←30→ プリントする場合，以下のサブルーチンを実行し，水平タブコードをプリンターへ送っておくこと 500 OUT &H10,&H9:OUT&H40,0:OUT &H40,1 510 D=INP(&H40) AND 1:IF D=0 THEN 520 ELSE 510 520 RETURN

機　能	命　令
水平タブを全面解除する	LPRINT　CHR$(27);CHR$(50);
水平タブを部分解除する	LPRINT　CHR$(27);CHR$(41);"a, b."; (例) 前ページの例のうち，10と20の水平タブを解除する LPRINT　CHR$(27);CHR$(41);"010,020.";
レフトマージンを設定する	LPRINT　CHR$(27);"L";"n"; nは3桁の10進数で記入する (例) LPRINT　CHR$(27);"L";"010"; ←ここからプリントする ←10→
レフトマージンを解除する	LPRINT　CHR$(27);"L000"; Lの後にゼロを3個書けばよい
n行改行する	LPRINT　CHR$(31);CHR$(16+n); 1=<n=<15のこと
ドットスペースをプリントする	LPRINT　CHR$(27);CHR$(n); 1=<n=<6のこと。 プロポーショナル文字のときのみ6ドットまでスペースがとれる

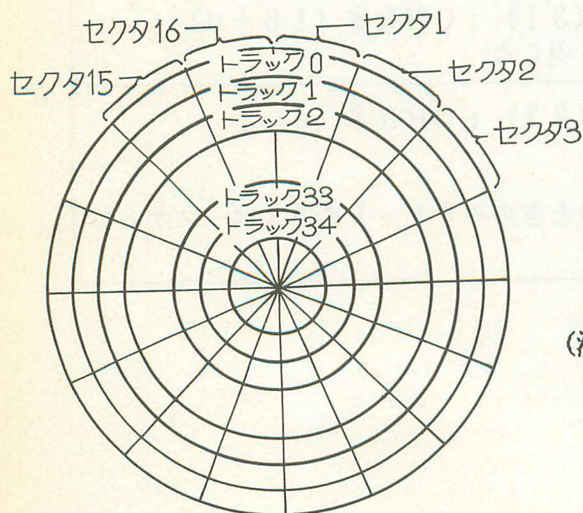
ミニフロッピーディスク

＜基本事項＞
ＰＣ－８００１の外部記憶装置としてミニフロッピーディスク（ＰＣ－８０３１，８０３２）がある。ＰＣ－８０３２は，ＰＣ－８０３１と接続して使う拡張ディスクユニットで，両方とも２台のミニフロッピーディスクドライブが入っている。ＰＣ－８００１とＰＣ－８０３１の接続には，拡張ユニット（ＰＣ－８０１１），Ｉ／Ｏユニット（ＰＣ－８０１２），Ｉ／Ｏポート（ＰＣ－８０３３）のいずれかを用いる。

＊フロッピーディスクの外形



＊ディスクの構造



１セクター＝２５６バイト（２５６文字）
８セクター＝１クラスター
２クラスター＝１トラック
３５トラック＝ディスク１枚

１６セクター×３５＝５６０セクター
５６０セクター×２５６＝１４３３６０バイト
(注)
ユーザーが使うことができるのは，
１３９２６４バイト。

＜ディスクファイルの管理＞

フロッピーディスクの読み書きは，１セクター単位で行われる。したがって，トラック番号とセクター番号で物理的に読み書きする場所が決められる。通常は，特別に意識しなくても，ファイル名さえ与えてやれば，自動的に読み書きする。管理は，１クラスター単位でＢＡＳＩＣが行っている。

項目の集まりをレコードといい，レコードの集まりをファイルという。

＜トラックの使用割り当て＞

＊システムディスク（ＤＩＳＫ　ＢＡＳＩＣが記憶されているディスク）

トラック	セクター	内　　容
０	１	ＩＰＬ
０	２～１６	ＤＩＳＫ　ＢＡＳＩＣ
１～２	すべて	
３～１７	すべて	ユーザーが使える領域
１８	１～１２	ディレクトリー
１８	１３	ＩＤ
１８	１４～１６	ＦＡＴ
１９～３４	すべて	ユーザーが使える領域

＊データディスク（ＤＩＳＫ　ＢＡＳＩＣを含まないディスク）

トラック	セクター	内　　容
０～１７	すべて	ユーザーが使える領域
１８	１～１２	ディレクトリー
１８	１３	ＩＤ
１８	１４～１６	ＦＡＴ
１９～３４	すべて	ユーザーが使える領域

<ディレクトリー>

ディレクトリーとは，ディスクに記憶されているファイルの名前，属性，どの場所から記憶されているかを示す登録簿のこと。ファイルごとに１つ作られ，１つのディレクトリーは１６バイトとられる。

ディレクトリーの構造

バイト	内　　　容
０～５	ファイル名　先頭バイトが１６進数のＦＦならば，未使用のディレクトリー　０ならば，削除されたファイル
６～８	拡張子
９	属性を示す 　０：アスキー形式 １０：アスキー形式，書き込み禁止 ４０：アスキー形式，リード・アフター・ライト ８０：非アスキー形式 ９０：非アスキー形式，書き込み禁止 Ｃ０：非アスキー形式，リード・アフター・ライト
１０	ファイルの先頭クラスター番号
１１～１５	未使用

<ＦＡＴ>

ＦＡＴ（ファイル・アロケーション・テーブル）は，各クラスターの使用・未使用状態を示すもの。ファイルは普通，物理的につながったクラスター上に作られるわけではなく，離れたクラスターをつないで構成される。ＭＯＵＮＴコマンドを実行すると，ＦＡＴがＰＣ本体のメモリーにコピーされる。ＦＡＴは，１４～１６の各セクターにとられ，それぞれのセクターの内容はすべて同一となっている。各セクターの０～６９バイトがそれぞれのクラスターに対応し，１バイトでクラスターの使用状態を表す。

ＦＡＴの値	クラスターの状態
０～４５	使用中を示す　後続クラスターの番号が入っている
Ｃ１～Ｃ８	使用中を示す　連続するクラスターの最後　下位４ビットには，そのクラスターで実際に使われているセクターの数が入っている
ＦＥ	使用できない　ＤＩＳＫＢＡＳＩＣやディレクトリー等で使っている
ＦＦ	未使用

<ＤＩＳＫ　ＢＡＳＩＣのスタート>

（1）　ディスクユニットの電源を入れる。
（2）　システムディスクをドライブ1に入れて，ふたをしめる。
（3）　ＰＣ－８００１の電源を入れる。
（4）　システムプログラムが読み込まれる。
（5）　画面に次の表示が出る。

```
DISK Version 〔14―NOV―1980〕
How many files (0―15) ?
```

（6）　同時にオープンできるファイルの数は１５までなので，１５以下の数字を入力すると，ＯＫと出て，準備完了。

<バックアップファイルの取り方>

ディスクに記憶されているファイルは，使用中のミスによりこわれてしまうことがある。重要なファイルが記憶されているディスクは，あらかじめ複写しておいたほうがよい。システムディスクは，ディスクユニットを買うと1枚しかついてこないので，特に必要。

＊システムディスクのバックアップ作成

（1）　ドライブ1にシステムディスクを装着し，ドライブ2にバックアップディスクを装着する。
（2）
```
mount1          (RET)
OK
run"backup"     (RET)
Back up a disk
Mount master disk on drive1, then
hit return
 (RET)
Mount new disk on drive2, then
hit return
 (RET)
Format disk (y/n) ?
```
ディスクのフォーマットが必要ならば，ｙ（ＲＥＴ）。
必要なければ，ｎ（ＲＥＴ）。
（3）　最後にＣｏｍｐｌｅｔｅとでて，完了。

(注)　（ＲＥＴ）は，ＲＥＴＵＲＮキーを押すことを示す。

データディスクのバックアップファイルも同様にして作ることができる。
（1）でシステムディスクのかわりにデータディスクを装着すればよい。

＜フォーマットの仕方＞

ミニフロッピーディスクをＰＣ－８０３１で使い，入出力ができるようにする作業をディスクのフォーマッティングという。

レベル	実行方法	内　　　容
1	ＦＯＲＭＡＴコマンドを使う	ＢＡＳＩＣのＤＳＫＩ＄，ＤＳＫＯ＄でのみ，読み書きできるようになる。ＦＡＴ，ディレクトリー，ＩＤの初期化はされないので，通常のファイルを作ることはできない
2	システムディスクのｆｏｒｍａｔプログラムを使う	ＦＡＴ，ディレクトリー，ＩＤの初期化をする ＢＡＳＩＣのコマンド，ステートメントを使って，ファイルを作ることができるようになる

＊ｆｏｒｍａｔプログラムの実行

(1) システムディスクからｆｏｒｍａｔプログラムをロードし，実行する。
(2) ｃｒｅａｔｅ　ｓｙｓｔｅｍ　ｄｉｓｋ　（ｙ／ｎ）？

　　システムディスクを作るならｙ（ＲＥＴ），データディスクを作るならｎ（ＲＥＴ）と答える。
(3) ｍｏｕｎｔ　ｎｅｗ　ｄｉｓｋ　ｏｎ　ｄｒｉｖｅ２
　　ｓｕｒｅ（ｙ／ｎ）？

　　フォーマッティングするディスクをドライブ２へ装着し，ｙ（ＲＥＴ）。
(4) フォーマッティングされる。データディスクを作るのであればこれで終了。
　　システムディスクを作るときは，さらに以下の表示が出る。
　　ｍｏｕｎｔ　ｓｙｓｔｅｍ　ｄｉｓｋ　ｏｎ　ｄｒｉｖｅ１
　　ｓｕｒｅ（ｙ／ｎ）？

　　システムディスクをドライブ１に装着し，ｙ（ＲＥＴ）。
(5) Ｃｏｍｐｌｅｔｅと出て，すべて終了。

＜ＤＩＳＫ　ＢＡＳＩＣのバージョンの違い＞

① ＤＩＳＫ　ＢＡＳＩＣの新しいバージョンは，スタート時に

　　Ｄｉｓｋ　Ｖｅｒｓｉｏｎ　〔１４―Ｎｏｖ―１９８０〕

　　と表示される。古いバージョンは［　　　］内の表示はない。

② 古いバージョンでは，メモリー上のＦＡＴの内容が変更されたか否かにかかわらず，ｒｅｍｏｖｅ時にＦＡＴをディスクに書き込む。
　　新しいバージョンでは，ＦＡＴの内容に変更があったときのみｒｅｍｏｖｅ時にＦＡＴをディスクに書き込む。

③　②の理由から，ライトプロテクトノッチに書き込み禁止のラベルをはったディスクに対してｒｅｍｏｖｅを行ったとき，以下のようになる。

a　古いバージョンでは，ＦＡＴを書き込もうとして書き込めないのでループ状態となる。つまり，ｒｅｍｏｖｅはできない。

b　新しいバージョンでは，ＦＡＴを書き込まないので，正常に終了する。

データファイル

項目（アイテム）の集まりをレコードといい，レコードの集まりをファイルという。

社員コード	社員名	住所	社員コード	社員名	住所	社員コード	社員名	

項目

レコード

ファイル

データファイルには，次の二種類がある。

①　シーケンシャルファイル

「順編成ファイル」とも呼ばれる。レコードを順次にならべていったファイル。１０番目にあるレコードを読みたいときでも，１番目のレコードから順番に読んでいかなければならない。また，ファイルの中にあるレコードの内容を削除，変更することはできない。削除，変更するときは，新しいファイルを作る必要がある。レコードの長さは一定ではない。カセットテープのファイルなどは，典型的なシーケンシャル・ファイル。

②　ランダムファイル

「乱編成ファイル」とも呼ばれる。レコード番号を指定すれば，読み書きはどのレコードからでもできる。ディスクでのみ，使うことができる。レコードの長さは，２５６バイトで一定。

<シーケンシャルファイル>

＊シーケンシャルファイルに必要な命令

命　　　令	機　　　　　　能
MOUNT	
OPEN	ファイルを開く モード　INPUT……読む APPEND……追加 OUTPUT……新規作成
PRINT#	レコードを書き出す
PRINT#　USING	書式を指定してレコードを書き出す
INPUT#	レコードを読む
LINEINPUT#	1行をすべて文字列として読む
INPUT$関数	指定した文字をファイルから読む
EOF関数	ファイルの終わりを検出する
CLOSE	ファイルを閉じる
REMOVE	

＊入出力命令

①　数字の入出力
<出力>

```
500　S=789
510　T=-56.7
520　U=123456
530　PRINT#1, S;T
540　PRINT#1, USING "######,";U
```

ディスクの記録のされ方



Uは書式指定して書き出したので，数字の間に"，"がはいる。"，"は，項目の区切りとみなされるので，2つの項目に分割される。
CR（キャリッジ・リターン）とLF（ライン・フィード）は，レコードの境界を表す文字で，それぞれ16進数で0Dと0A。

＜入力＞

前記のファイルを読む場合

```
100  INPUT#1, S, T
110  INPUT#1, U1, U2
120  PRINT S;T;U1;U2
```

結果

```
   789  -56.7   123   456
```

② 文字の入出力

```
200  S$="マイコン"
210  T$="PC8001"
220  PRINT#1, S$;T$
230  PRINT#1, S$;", ";T$
240  PRINT#1, CHR$(34);"マイコン    , PC8001
     ";CHR$(34)
```

*ディスクの記録のされ方

マ	イ	コ	ン	P	C	8	0	0	1	CR	LF	マ	イ	コ	ン	,

P	C	8	0	0	1	CR	LF	”	マ	イ	コ	ン			,	P

C	8	0	0	1	”	CR	LF

文字データの場合，項目の区切りとなるのは空白ではなくコンマ（，）である。
文字データの中にコンマまたは，意味のある空白を入れたい場合は，CHR$（34）（”）で囲む。

前記のファイルを以下のようにして読む。最初のレコードは，S$とT$がつながって1つの項目となる。（ST$として読む）

```
100  INPUT#1, ST$
110  INPUT#1, S$, T$:INPUT#1, U$
120  PRINT ST$:PRINT S$, T$
130  PRINT U$
```

結果

```
マイコンPC8001
マイコン                PC8001
マイコン    , PC8001
```

<シーケンシャルファイルのプログラム例１>

＊機能

シーケンシャルファイル”セイセキ”を作成する。入力した番号，名前，点数で１レコード。入力終了時は，番号を９９９９とする。

＊ポイント

文字項目を書き出すときは，間にコンマ（，）を入れる。

```
 10 '**********************
 20 '*                    *
 30 '*  シーケンシャル レイ (1)  *
 40 '*                    *
 50 '**********************
 60 OPEN "セイセキ" FOR OUTPUT AS #1
 70 INPUT "バンゴウ" ;B$
 80 IF B$="9999" GOTO 130
 90 INPUT "ナマエ";N$
100 INPUT "テンスウ";T
110 PRINT#1, B$;", ";N$;",";T
120 GOTO 70
130 CLOSE#1
140 END
```

<シーケンシャルファイルのプログラム例２>

＊機能

例（1）で作成したファイルを読み，画面上に番号，名前，点数を表示する。

＊ポイント

ＥＯＦ（1）でファイルの終了を判断する。

```
 10 '**********************
 20 '*                    *
 30 '*  シーケンシャル レイ (2)  *
 40 '*                    *
 50 '**********************
 60 OPEN "セイセキ" FOR INPUT AS#1
 70 IF EOF(1) GOTO 110
 80 INPUT#1, B$, N$, T
 90 PRINT B$, N$, T
100 GOTO 70
110 CLOSE#1
120 END
```

＜シーケンシャルファイルのプログラム例3＞

＊機能

例（1）で作成したファイルにデータを追加する。追加データが終わりのときは，番号を9999とする。

＊ポイント

シーケンシャルファイルにレコードを追加するときのモードは，ＡＰＰＥＮＤ。

```
 10 '＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
 20 '＊                                       ＊
 30 '＊    シーケンシャル レイ   (3)          ＊
 40 '＊                                       ＊
 50 '＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
 60 OPEN "セイセキ" FOR APPEND AS #1
 70 INPUT "バンゴウ" ;B$
 80 IF B$="9999" GOTO 130
 90 INPUT "ナマエ";N$
100 INPUT "テンスウ";T
110 PRINT#1,B$;",";N$;",";T
120 GOTO 70
130 CLOSE#1
140 END
```

＜ランダムファイル＞

＊ランダムファイルに必要な命令

命　　　令	機　　　能
MOUNT	
OPEN	ファイルを開く
GET	ディスクからバッファへ読み込む
PUT	バッファからディスクへ書き出す
FIELD	バッファの項目を定義する
LSET	バッファへデータを入れる
RSET	（同上）
MKI$関数	数字を文字へ変換する
MKS$関数	（同上）
MKD$関数	（同上）
CVI関数	文字を数字へ変換する
CVS関数	（同上）
CVD関数	（同上）
LOF関数	最大のレコード番号を与える
LOC関数	PUTまたはGETしたレコード番号を与える
CLOSE	ファイルを閉じる
REMOVE	

＜バッファについて＞

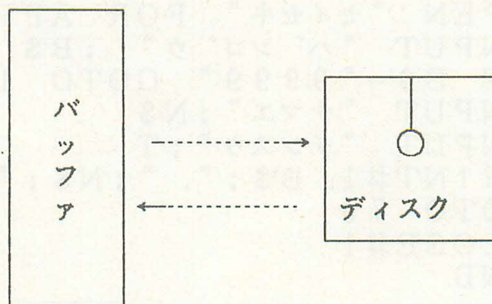
ランダムファイルの場合，ディスクとの入出力にバッファという作業領域を必要とする。バッファは，ランダムファイルの1レコードの長さに等しい２５６バイトで，１５個まで使うことができる。ＤＩＳＫ　ＢＡＳＩＣを開始する際，Ｈｏｗ　ｍａｎｙ　ｆｉｌｅｓ　（０－１５）？　という問いには，同時にオープンするファイルの数を答える。この入力によって使うことのできるバッファの数が決まる。

ＯＰＥＮ，ＧＥＴ等で指定するファイル番号によって，使用するバッファの番号が決まる。バッファ内の項目の定義は，ＦＩＥＬＤ文でおこなう。ランダムファイルの場合，入出力はすべて文字項目として扱う。数字項目を書き出すときは，ＭＫＳ＄，ＭＫＩ＄，ＭＫＤ＄で文字項目に変換してバッファへ入れる。数字項目を読み込むときは，ＣＶＳ，ＣＶＩ，ＣＶＤで数字項目に変換して使う。

（数値変数）
ＭＫＩ＄，ＭＫＳ＄，ＭＫＤ＄
ＬＳＥＴ，ＲＳＥＴ

ＣＶＩ，ＣＶＳ，ＣＶＤ

（文字変数）
ＬＳＥＴ，ＲＳＥＴ



＜ランダムファイルプログラム例（１）＞

＊機能

ランダムファイル”ウリアゲ”を作成する。入力項目は，社員コード，名前，商品Ａ・Ｂ・Ｃそれぞれの売り上げ金額。入力終了のときは，社員コードを９９９９９９と入れる。

＊ポイント

数字は，ＭＫＳ＄（ＭＫシリーズ関数）で文字項目に変換し，バッファへ送る。

```
 10 '***  ランダム レイ (1)  ***
 20 OPEN "ウリアゲ" AS #1
 30 FIELD #1, 6 AS X$, 14 AS Y$, 4 AS A$,
          4 AS B$, 4 AS C$
 40 INPUT "シャインコード";P$
 50 IF P$="999999" GOTO 130
 60 INPUT "ナマエ";Q$
 70 INPUT "A ノ ウリアゲ";A
 80 INPUT "B ノ ウリアゲ";B
 90 INPUT "C ノ ウリアゲ";C
100 LSET X$=P$:LSET Y$=Q$
110 LSET A$=MKS$(A):LSET B$=MKS$(B):
    LSET C$=MKS$(C)
120 PUT#1:GOTO 40
130 CLOSE#1:END
```

<ランダムファイルプログラム例（2）>

＊機能
例（1）で作成したファイルを読み，指定した社員コードの売り上げを画面上に出力する。処理終了のときは，社員コードを９９９９９９と入力する。
＊ポイント
レコードナンバーと社員コードの対応をとるため，あらかじめすべてのレコードを読んで対応表をつくっている。

```
 10 '***************
 20 '* ランダ゛ム レイ(2) *
 30 '***************
 40 OPEN "ウリアケ゛" AS #1
 50 FIELD#1,6 AS X$,14 AS Y$,4 AS A$,
         4 AS B$, 4 AS C$
 60 DIM N(10),M$(10)
 70 FOR I=1 TO LOF(1)
 80   GET#1
 90   N(I)=LOC(1):M$(I)=X$
100 NEXT I
110 INPUT "シャイン コート゛";S$
120 IF S$="999999" GOTO 220
130 FOR I=1 TO LOF(1)
140   IF M$(I)=S$ GOTO 170
150 NEXT I
160 PRINT "カ゛イトウ シャインコート゛ ナシ!":GOTO 110
170 GET#1,N(I)
180 PRINT S$;
190 PRINTUSING"&                &";Y$;
200 PRINTUSING"######## ######## ########"
    ;CVS(A$);CVS(B$);CVS(C$)
210 GOTO 110
220 CLOSE#1
230 END
```

＊特殊なＩ／Ｏ

いままで説明してきた命令によるファイルの入出力は，ＦＡＴによって管理される。ＤＳＫＩ＄，ＤＳＫＯ＄は，ＦＡＴとは関係無しにドライブ番号，トラック番号，セクター番号を指定することにより，ディスク上の任意の場所に対して入出力することができる。マウント命令も必要ない。

任意の場所の内容を読む……ＤＳＫＩ＄
任意の場所に書き出す………ＤＳＫＯ＄

ＤＳＫＩ＄，ＤＳＫＯ＄の入出力は，０番のバッファを使用する。
使い方を誤るとディスク上のファイルを壊してしまうことがあるので，使うときは，十分な注意が必要。

<DSKI$のプログラム例>

*機能

ディスクの使用状況をプリンターに出力する。ディスクを装着して，ドライブ番号に答えるだけでよい。MOUNT命令は必要ない。

*ポイント

行番号100～120までで，FIELD文を配列定義している。

DSKI$でディレクトリー，FATを読んでいる。バッファの番号は0。

```
10 '************************
20 '* デ゛ィスク カンリヒョウ サクセイ *
30 '************************
40 CLEAR1000
50 DIM N$(15), Z$(15), C$(15)
60 LPRINT "デ゛ィスク カンリヒョウ":LPRINT
70 INPUT "ド゛ライブ゛ ナンバ゛ーハ";D
80 LPRINT "ド゛ライブ゛ ナンバ゛ー ハ?";D
90 FAT$=DSKI$(D, 18, 14)
100 FOR I=0 TO 15
110   FIELD#0, I*16 AS DM$, 9 AS N$(I),
              1 AS Z$(I), 1 AS C$(I)
120 NEXT I
130 LPRINT "ファイルメイ    ソ゛クセイ  ショウクラスター"
140 FOR I=1 TO 12
150  DM$=DSKI$(D, 18, I)
160  FOR J=0 TO 15
170   IF ASC(N$(J))=0 OR
         ASC(N$(J))=&HFF THEN 270
180    Z=ASC(Z$(J)):C=ASC(C$(J))
190    F$=""
200    IF (Z AND &H80)=0 THEN F$=F$+"A"
210    IF (Z AND &H10)>0 THEN F$=F$+"P"
220    IF (Z AND &H40)>0 THEN F$=F$+"R"
230    LPRINT USING"&        & &&    ##"
              ;N$(J), F$, C;
240    C=ASC(MID$(FAT$, C+1))
250    IF C<=69 THEN LPRINT USING"->##"
                   ;C;:GOTO 240
260    LPRINT
270 NEXT J, I
280 FOR C=0 TO 69
290  S=ASC(MID$(FAT$, C+1))
300  IF S=&HFF THEN FR=FR+1
310 NEXT C
320 LPRINT "ノコリ ハ ";FR;"クラスター";
           FR*8*256;" バ゛イト"
330 END
```

エラーメッセージ

エラーコード	エラーメッセージ
1	NEXT without FOR FORがない。NEXTとFORが対応していない。
2	Syntax error 文法違反。LIST.で調べよ。
3	RETURN without GOSUB GOSUBとRETURNが対応していない。RETURNだけある。
4	Out of DATA READ文で読むべきデータがDATA文にない。
5	Illegal function call ステートメントの使い方がまちがっている。あるいは限度を越えている。
6	Overflow 数値や演算結果が許容範囲を越えた。
7	Out of memory プログラムや配列が大きすぎてメモリー不足。
8	Undefined line number プログラム行が定義されていない。
9	Subscript out of range 配列変数の添字が規定範囲にない。
10	Redimensioned array 同じ名前の配列を再定義した。
11	Division by zero 0で割った。あるいは，未定義の変数で割った。
12	Illegal direct ダイレクトモードで使えないコマンドが入った。
13	Type mismatch 式の左右の変数の型が合っていない。
14	Out of string space ストリング領域不足。CLEAR文を調べよ。

15 String too long

引用符（"　"）で囲まれた文字が２５５文字を越えた。

16 String formula too complex

文字式が複雑すぎる。

17 Can't continue

エラー停止かブレーク後，プログラムを変更したか，あるいはプログラムが存在しないためＣＯＮＴできない。

18 Undefined user function

ＤＥＦ文で定義する前にユーザー関数を呼んだ。

19 No RESUME

エラー処理に入ったが，ＲＥＳＵＭＥがない。

20 RESUME without error

エラーがないのにＲＥＳＵＭＥがある。

21 Unprintable error

エラーメッセージに定義されていないエラー。通常，未定義のエラーコードを含むＥＲＲＯＲ文で起こる。

22 Missing operand

ステートメントの中に必要となるパラメーターなどがない。

23 Line buffer overflow

行を入力するとき，長さの有効範囲を超えた。通常は超えた部分は無視され，エラーメッセージは出ない。

24 Position not on screen

指定したカーソル位置などが，画面の範囲外。

25 Bad File Data

ファイルにあるデータの形式がまちがっている。

26 Disk BASIC Feature

ディスクがつながっていないのに，Ｄｉｓｋ ＢＡＳＩＣの命令を実行。

27 Communications Buffer Overflow

周辺機器との入出力のためのバッファがオーバーフローした。

28 Port not initialized

インタフェース用ＬＳＩの初期設定がされていない。

29　Ｔａｐｅ　ｒｅａｄ　ＥＲＲＯＲ
カセットテープがうまく読み取れない。

＜ＤＩＳＫ　ＢＡＳＩＣのエラー＞

50　ＦＩＥＬＤ　ｏｖｅｒｆｌｏｗ
ＦＩＥＬＤ文で２５７バイト以上の文字を割り当てた。

51　Ｉｎｔｅｒｎａｌ　ｅｒｒｏｒ
ＢＡＳＩＣ内部でのエラー。

52　Ｂａｄ　ｆｉｌｅ　ｎｕｍｂｅｒ
オープンされていないファイル・ナンバーをアクセスした。

53　Ｆｉｌｅ　ｎｏｔ　ｆｏｕｎｄ
ＬＯＡＤ，ＫＩＬＬ，ＯＰＥＮなどで存在しないファイルをアクセスした。

54　Ｂａｄ　ｆｉｌｅ　ｍｏｄｅ
シーケンシャル・ファイルでＯＰＥＮしたファイルにランダム・アクセスした。

55　Ｆｉｌｅ　ａｌｒｅａｄｙ　ｏｐｅｎ
すでに開かれているファイルにＯＰＥＮやＫＩＬＬした。

56　Ｄｉｓｋ　ｎｏｔ　ｍｏｕｎｔｅｄ
ｍｏｕｎｔしていないディスクをアクセスした。

57　Ｄｉｓｋ　Ｉ／Ｏ　ｅｒｒｏｒ
ディスクにＲＥＡＤ・ＷＲＩＴＥｅｒｒｏｒが生じ，読み直しても修正できなくなった。

58　Ｆｉｌｅ　ａｌｒｅａｄｙ　ｅｘｉｓｔｓ
ＮＡＭＥ文によって定義されたファイル名がすでに登録されている。

59　Ｄｉｓｋ　ａｌｒｅａｄｙ　ｍｏｕｎｔｅｄ
ｍｏｕｎｔされているディスクに対してまたｍｏｕｎｔした。

60　Ｄｉｓｋ　ｆｕｌｌ
ディスク上のすべての場所を使いきった。

61　Ｉｎｐｕｔ　ｐａｓｔ　ｅｎｄ
ファイルのすべてのデータを読みつくした後にｉｎｐｕｔ文が実行された。

62　Ｂａｄ　ｆｉｌｅ　ｎａｍｅ
不適当なファイルネームを使った。

63　Ｄｉｒｅｃｔ　ｓｔａｔｅｍｅｎｔ　ｉｎ　ｆｉｌｅ
アスキー・フォーマットでのＬＯＡＤ中にダイレクト・ステートメントがあった。

64　Ｂａｄ　ａｌｌｏｃａｔｉｏｎ　ｔａｂｌｅ
ＦＡＴがこわれた。

65　Ｂａｄ　ｄｒｉｖｅ　ｎｕｍｂｅｒ
不適当なドライブネームを指定した。

68　Ｒｅｎａｍｅ　ａｃｒｏｓｓ　ｄｉｓｋｓ
ファイル名のつけかえで，違ったドライブナンバーを指定した。

71　Ｆｉｌｅ　ｎｏｔ　ＯＰＥＮ
まだ開かれていないファイルに対してアクセスした。

72　Ｆｉｌｅ　ｗｒｉｔｅ　ｐｒｏｔｅｃｔｅｄ
ライト・プロテクトがかかっているファイルに対して書き込んだ。

ＰＣ－８０００シリーズ　ラインアップ

型　　　　名	品　　　　名	特　　　　長
ＰＣ－８００１	パーソナルコンピューター	８ビット，パーソナルコンピューター
ＰＣ－８００６	増設ＲＡＭ	８００１，８０１１の増設用ＩＣセット
ＰＣ－８０１１	拡張ユニット	ＰＲＯＭ，ＲＡＭ，ＲＳ－２３２Ｃインタフェース，パラレルポート，リアルタイムクロック，ミニフロッピーディスクインタフェースを備えた拡張ユニット
ＰＣ－８０１２	Ｉ／Ｏユニット	リアルタイムクロック，ミニフロッピーディスクインタフェース，各種拡張用ボード差し込み用ソケットを備えたユニット
ＰＣ－８０１２—０１	８０１２用ユニバーサルボード	自作用プリントボード
ＰＣ－８０２３	８０桁ドットマトリックスプリンター	多様な機能を持った，ＣＰＵ内蔵のプリンター　ドットグラフィック，ひらがなが使用できる
ＰＣ－８０３１	ミニディスクユニット	ＣＰＵ内蔵のインテリジェントミニフロッピーユニット。１４３Ｋバイト×２ドライブ実装
ＰＣ－８０３２	ミニディスク拡張ユニット	ＰＣ－８０３１の拡張用ドライブユニット　１４３Ｋバイト×２ドライブ実装
ＰＣ－８０３３	ＰＣ－８０３１ＦＤＣ Ｉ／Ｏポート	ＰＣ－８０３１とＰＣ－８００１を，ＰＣ－８０１１，ＰＣ－８０１２なしで接続するためのＩ／Ｏユニット
ＰＣ－８０３１－２Ｗ	両面倍密デュアルミニディスクユニット	両面倍密度タイプ ３２０Ｋバイト×２ドライブ実装
ＰＣ－８０３２－２Ｗ	拡張用両面倍密デュアルミニディスクユニット	ＰＣ－８０３１－２Ｗの拡張用ドライブユニット ３２０Ｋバイト×２ドライブ実装

ＰＣ－８０００シリーズ　ラインアップ

型　　名	品　　名	特　　長
ＰＣ－８０３１－１Ｖ	シングルドライブ片面ミニディスクユニット	ＰＣ－８０３１の１ドライブタイプ。１４３Ｋバイト×１ドライブ
ＰＣ－８０３１－ＦＤ１	拡張用シングルドライブユニット	ＰＣ－８０３１－１Ｖの拡張用ドライブユニット
ＰＣ－８０４１	１２インチ・グリーンディスプレー	
ＰＣ－８０４２	１２インチ・カラーディスプレー	低解像度
ＰＣ－８０４４	家庭テレビ用カラーアダプター	家庭用カラーテレビをＰＣ－８００１に接続するときに使用
ＰＣ－８０４５	ライトペン	
ＰＣ－８０４６	９インチ・グリーンディスプレー	
ＰＣ－８０４９	１２インチ・カラーディスプレー	高解像度
ＰＣ－８０６２	ＲＳ－２３２Ｃケーブル	ＰＣ－８００１をモデム・音響カプラーに接続するときに使用
ＰＣ－８０９１	カラーディスプレー用ケーブル	ＰＣ－８００１とＰＣ－８０４９を接続するときに使用
ＰＣ－８０９２	グリーンディスプレー用ケーブル	ＰＣ－８００１に付属
ＰＣ－８０９３	ＣＭＴ用ケーブル	ＰＣ－８００１に付属
ＰＣ－８０９４	プリンター用ケーブル	ＰＣ－８００１とプリンターを接続するときに使用
ＰＣ－８０９５	ＰＣ－８０１１用ＲＳ－２３２Ｃケーブル	ＰＣ－８０１１をモデム・音響カプラーに接続するときに使用
ＰＣ－８０９６	ＰＣ－８０１１用ＩＥＥＥ－４８８ケーブル	ＰＣ－８０１１をＩＥＥＥ－４８８規格の装置に接続するときに使用

ＰＣ－８０００シリーズ ラインアップ

型　　名	品　　名	特　　長
ＰＣ－８０９７	ＧＰ－ＩＢ（ＩＥＥＥ－４８８）インタフェースセット	コントロール用ファームウエア＋ドライブ用バッファ＋ＰＣ－８０９６のセット
ＲＭ－２１０	カセットテープレコーダー	プログラムのセーブとロード用

ＰＣ－８０００シリーズには，拡張用のユニットとして，ＰＣ－８０１１とＰＣ－８０１２の２種類が用意されている。ＰＣ－８０１１は各種のインターフェースが内蔵されたもので，ケーブルさえつなげばすぐに各装置に接続することができる。

ＰＣ－８０１２は，最小限の機能におさえ，必要に応じてボードを差し込んで使うよう，ソケットが７枚分内蔵されている。

＜ＰＣ－８０１１の機能＞

1	３２ＫバイトＲＡＭ	ＣＰ／Ｍ，ＵＣＳＤ ＰＡＳＣＡＬなどを使用するとき，Ｎ－ＢＡＳＩＣのＲＯＭを切り離し，こちらのＲＡＭを使用する
2	８ＫバイトＰＲＯＭ	ユーザーのプログラムをＲＯＭ化して差し込むことができる
3	ＲＳ－２３２Ｃインタフェース	ＥＩＡ（Electoronic Industrial Association）の決めたシリアルデータの伝送規格に対応したインタフェースで，モデム・音響カプラーが接続できる
4	ＦＤＣ Ｉ／Ｏポート	ミニディスクユニットを接続するためのＩ／Ｏポート
5	汎用（はんよう）パラレルＩ／Ｏポート	８ビット入力ポート，８ビット出力ポート，４ビット入力ポート，４ビット出力ポートがあり，各種入出力機器を接続できる
6	ＩＥＥＥ－４８８インタフェース	ＩＥＥＥ－４８８インタフェースバスは，計測器とコンピューターを接続するインタフェースである
7	拡張Ｉ／Ｏバス	ユーザーに開放されたＩ／Ｏ命令実行時の入出力機能と割り込み処理に必要な諸データの入出力機能
8	割り込み制御回路	１６レベルプライオリティー付の割り込みコントロール機能
9	リアルタイム割り込み	１．６６msの周期でＣＰＵに割り込みをかける機能

PC－8000シリーズ ラインアップ

＜PC－8012の機能＞

1	マザーバス	ユーザーが開発したインタフェースカードを挿入するためのソケットが7つある。増設RAMボードが4枚（128Kバイト）まで挿入できる。
2	PROM 2Kバイト	ユーザーのプログラムをROM化して差し込むことができる。
3	FDC I／Oポート	ミニディスクユニットを接続するためのI／Oポート
4	割り込み制御回路	8レベル・プライオリティー付の割り込みコントロール機能。
5	リアルタイム割り込み	1．66msの周期でCPUに割り込みをかける機能

＜PC－8012－02　増設RAMボード＞

32KバイトのRAMを持ち，PC－8001のマスクROM部分を禁止し，こちらのRAMをアクセスできる。このボードは，バンク切り替えにより4枚まで使用できる。ボードごとにリード制御とライト制御ができ，2つ以上のバンクに同時にデータを書き込むことができる。

＜PC－8012－03　音声認識ボード＞

あらかじめ登録された話者の話す60語を，認識率98％で音声入力できるもの。プログラム，データの入力の他，ゲーム，ビジネスプログラムの実行を音声で指示できる。

（例）　PC－8011へのメモリー増設（BASICエリア拡張）

```
10 DEFUSR0=&HFF80
20 FOR J=0 TO 15
30 READ A$:POKE&HFF80+J,VAL("&H"+A$)
40 NEXT J
50 A=USR(0)
60 DATA 01,00,60,11,00,00,21,00
70 DATA 00,ED,B0,D3,E2,C3,E9,17
80 END
```

PC－8011へRAM32Kバイトを増設した場合，PC本体のRAM32Kバイトと合わせて64Kバイトとなるが，8011側へBASIC24Kバイト分をとっているので，実質的にはユーザーエリアは40Kバイトにしかならない。

(例)　PC－８０１２へのメモリー増設（BASICエリア拡張）

```
10 DEFUSR0=&HFF80
20 FOR J=0 TO 25
30 READ A$:POKE&HFF80+J,VAL("&H"+A$)
40 NEXT J
50 A=USR(0)
60 DATA 3E,01,D3,E7,3E,10,D3,E2,01,00
70 DATA 60,11,00,00,21,00,00,ED,B0,3E
80 DATA 11,D3,E2,C3,E9,17
90 END
```

これも８０１１の場合と同じで実質的には８Kバイトしか増設されない。



これが次のようになる。



ユーザーが使える領域

もし，この２４Kバイトもユーザーが使うとすれば，N－BASICは動かないので，すべて機械語でプログラムを書かねばならない。BANKを切り替えながら使うことになる。

＜BANK切り替えの仕方＞

PCのアドレスの下位３２Kバイトは，I／O命令でそっくり切り替えることができる。これをバンク切り替えという。
PCでは下位３２KバイトにN－BASICのROMがあるので，バンク切り替えは，機械語でやらなければならない。

PC－８０１１の場合　………E２H番地に対するOUT命令を実行する。出力するデータの値は問わない。

PC－８０１２の場合　………バンク０のみ使用した場合，E２H番地に次のデータを出力する。

書き込みのみ許可….１０H
読み出しのみ許可….０１H
読み書きとも許可….１１H
読み書きとも禁止….００H

カッコ内のすうじ

もじ

キャラクター・コード表

16進数は横が上位4ビット縦が下位4ビット マスの中の数字は16進数に対応する10進数 たとえば4BはKだが10進では75になる

16進	0	1	2	3	4	5	6	7
0	0	DE 16	32	0 48	@ 64	P 80	96	p 112
1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
2	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
3	EX 3	D3 19	# 35	3 51	C 67	S 83	c 99	s 115
4	ET 4	D4 20	$ 36	4 52	D 68	T 84	d 100	t 116
5	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
6	AK 6	SN 22	& 38	6 54	F 70	V 86	f 102	v 118
7	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
8	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
9	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
A	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
B	HM 11	EC 27	+ 43	; 59	K 75	[91	k 107	{ 123
C	CL 12	→ 28	, 44	< 60	L 76	¥ 92	l 108	¦ 124
D	CR 13	← 29	— 45	= 61	M 77	] 93	m 109	} 125
E	SO 14	↑ 30	. 46	> 62	N 78	^ 94	n 110	~ 126
F	SI 15	↓ 31	／ 47	? 63	O 79	_ 95	o 111	127

特殊文字

(読み出したら、所定のことを実行)

(例) CHR$(11) → HOME , CHR$(13) → RETURN

8	9	A	B	C	D	E	F	16進
▁ 128	┴ 144	160	ー 176	タ 192	ミ 208	═ 224	╳ 240	0
▂ 129	┬ 145	。 161	ア 177	チ 193	ム 209	╞ 225	円 241	1
▃ 130	┤ 146	「 162	イ 178	ツ 194	メ 210	╪ 226	年 242	2
▄ 131	├ 147	」 163	ウ 179	テ 195	モ 211	╡ 227	月 243	3
▅ 132	▔ 148	、 164	エ 180	ト 196	ヤ 212	◢ 228	日 244	4
▆ 133	─ 149	・ 165	オ 181	ナ 197	ユ 213	◣ 229	時 245	5
▇ 134	│ 150	ヲ 166	カ 182	ニ 198	ヨ 214	◥ 230	分 246	6
█ 135	▕ 151	ァ 167	キ 183	ヌ 199	ラ 215	◤ 231	秒 247	7
▏ 136	┌ 152	ィ 168	ク 184	ネ 200	リ 216	♠ 232	248	8
▎ 137	┐ 153	ゥ 169	ケ 185	ノ 201	ル 217	♥ 233	249	9
▍ 138	└ 154	ェ 170	コ 186	ハ 202	レ 218	♦ 234	250	A
▌ 139	┘ 155	ォ 171	サ 187	ヒ 203	ロ 219	♣ 235	251	B
▋ 140	╭ 156	ャ 172	シ 188	フ 204	ワ 220	● 236	252	C
▊ 141	╮ 157	ュ 173	ス 189	ヘ 205	ン 221	○ 237	253	D
▉ 142	╰ 158	ョ 174	セ 190	ホ 206	゛ 222	╱ 238	254	E
┼ 143	╯ 159	ッ 175	ソ 191	マ 207	゜ 223	╲ 239	255	F

PRINT CHR$（n）の使い方

通常は，キャラクターコード表の文字（図形）が表示される。

例外表

n（16進）	コード表の文字	機能
07	BL	BEEPと同じ
09	HT	水平タブ
0A	LF	ラインフィード
0B	HM	カーソルをホームポジションに戻す
0C	CL	画面クリア
0D	CR	キャリッジリターン
1C	→	カーソル移動右
1D	←	カーソル移動左
1E	↑	カーソル移動上
1F	↓	カーソル移動下

キーボードからINKEY$で入力できる制御コード表

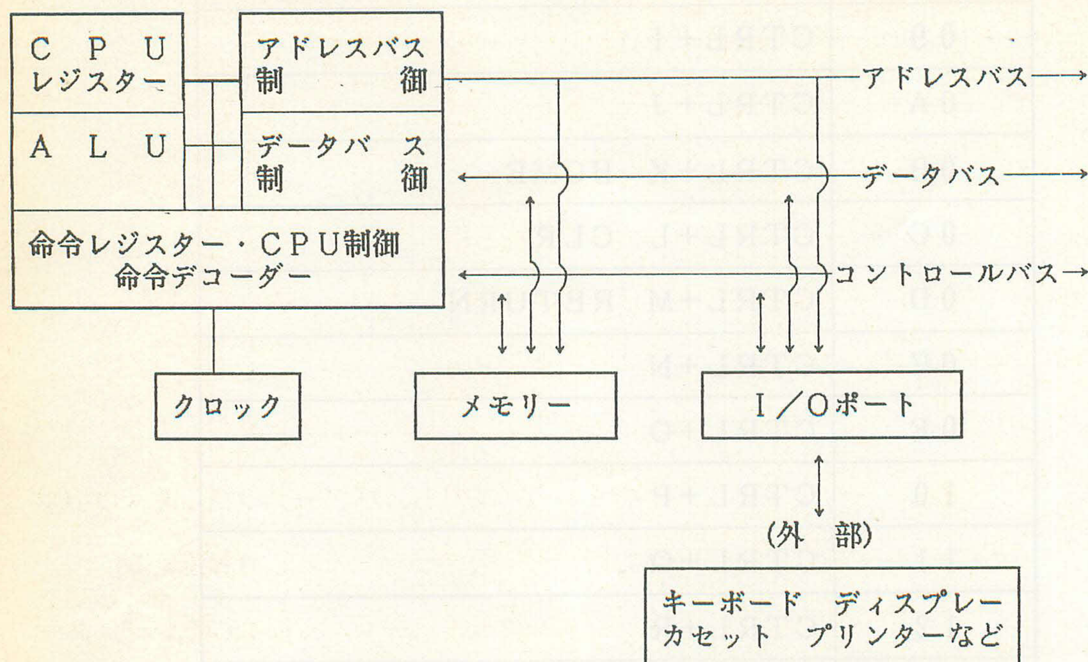
16進コード	キー
00	CTRL+@
01	CTRL+A
02	CTRL+B
03	CTRL+C
04	CTRL+D
05	CTRL+E
06	CTRL+F
07	CTRL+G

（例）

```
10  X$=INKEY$
20  IF  X$=CHR$ (&H01)
      THEN  GOTO  100
          ・
          ・
          ・
100  REM***PRG1***
```

上例では，CTRLキーとAのキーが押されたら，100行へとぶ。

16進コード	キー
08	CTRL+H　DEL
09	CTRL+I
0A	CTRL+J
0B	CTRL+K　HOME
0C	CTRL+L　CLR
0D	CTRL+M　RETURN
0E	CTRL+N
0F	CTRL+O
10	CTRL+P
11	CTRL+Q
12	CTRL+R
13	CTRL+S
14	CTRL+T
15	CTRL+U
16	CTRL+V
17	CTRL+W
18	CTRL+X
19	CTRL+Y
1A	CTRL+Z
1C	→
1D	←
1E	↑
1F	↓



CPU……………算術演算，論理演算，各種判断，制御などを行う。CPUは，時計のきざみ（クロックパルス）を処理単位として働く。
メモリー…………プログラムやデータの記憶を1バイト（8ビット）単位でする。メモリーには0～65535（&H0000～&HFFFF）まで2バイトで表されるアドレス（番地）がついている。
I／Oポート………CPUと外部との仲介役をする。いわばデータの出入り口。
アドレスバス………アドレス信号の伝送部。A0～A15まで16本の線（16ビット）の束。したがって，パラレル（並列的）に信号は伝送される。
データバス…………CPU，メモリー，I／Oポート間のデータ伝送路。D0～D7まで8本の線（8ビット）の束。
コントロールバス…読み込み，書き出しなど，CPUがメモリーやI／Oポートをどう動かすか指示する伝送路。
命令レジスター……データバスから読み込んだOPコード（オペコード，これが機械語）を書きこむ場所。ここへとりこむ動作を命令フェッチサイクルという。命令フェッチサイクルにあれば，データバス上のデータは無条件に命令レジスターに書かれてオペコードと解釈される。
命令デコーダー……オペコードの解釈をする。
CPU制御…………オペコードの内容や外部からの制御信号によって，CPUの各ブロックをどんな順序で，どう動かすかを判定して実行する。
ALU……………Arithmetic Logical Unit，演算論理機構。ここで計算や論理演算をする。
CPUレジスター…プログラム実行中のデータの一時記憶場所。

ＣＰＵレジスター

主レジスター		補助レジスター	
A	F	A'	F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

専用レジスター	
I	R
IX	
IY	
SP	
PC	

8ビット　16ビット

Ａレジスター……アキュムレーターという。演算およびデータの読み書きの中継点。
Ｂレジスター……①データの一時記憶　②減数カウンターとしても使える。
Ｃレジスター……①データの一時記憶　②Ｉ／Ｏポート指定レジスターとしても使う。Ｂ，Ｃレジスターはペア（一対）にしてＢＣレジスターとして使える。
Ｄ，Ｅレジスター…データの一時記憶場所。ＤＥペアでも使える。
Ｈ，Ｌレジスター…Ｄ，Ｅと同じ。ＨＬペアのアドレス指定命令は豊富にある。

Ｆレジスター……演算結果に応じて，変化するフラグ（ビットが１か０かで示す）をたてる。

7	6	5	4	3	2	1	0（ビット位置）
S	Z		H		P／V	N	C

Ｆレジスターの中身

Ｓ……サインフラグ。結果が正か負かを記憶する。
Ｚ……ゼロフラグ。
　１…結果がゼロ。
　０…結果がゼロでない。
Ｈ……ハーフキャリーフラグ。演算の結果，３ビット目から４ビット目にキャリー（桁上がり）があったかどうか記憶。
Ｐ／Ｖ…パリティー／オーバーフローフラグ。結果１が偶数個あるか奇数個あるかを記憶する。また，結果がオーバーフローしたかどうかを記憶。
Ｎ……加減算フラグ。演算が加算か減算かを記憶。
Ｃ……キャリーフラグ。ＣＹとも書く。演算の結果，キャリー（桁上がり）が生じたかどうかを記憶。

補助レジスター……‥この内容を直接操作する命令群はない。主レジスターと内容を交換して使う。

Ｉレジスター…………Ｚ８０の割り込みモード２のときに使う。

Ｒレジスター…………外部にダイナミックＲＡＭを接続した場合，記憶が消えないようにリフレッシュするときに使う。

ＩＸ，ＩＹレジスター‥インデックスレジスターとして使う。データのテーブル（表）を操作するときに便利。

ＳＰレジスター………スタックポインター。メモリーのアドレスを意識せずにレジスターからあるメモリーにデータを出し入れできる。ＳＰで指定するメモリーをスタックという。そのアタマを指す。プログラムであらかじめスタック領域を定め，その最高位のアドレス＋１をＳＰに入れておく必要がある。

ＰＣレジスター………プログラムカウンター。メモリーのプログラム領域から，命令を読み出すと＋１され，次の命令を順次に読み出す。

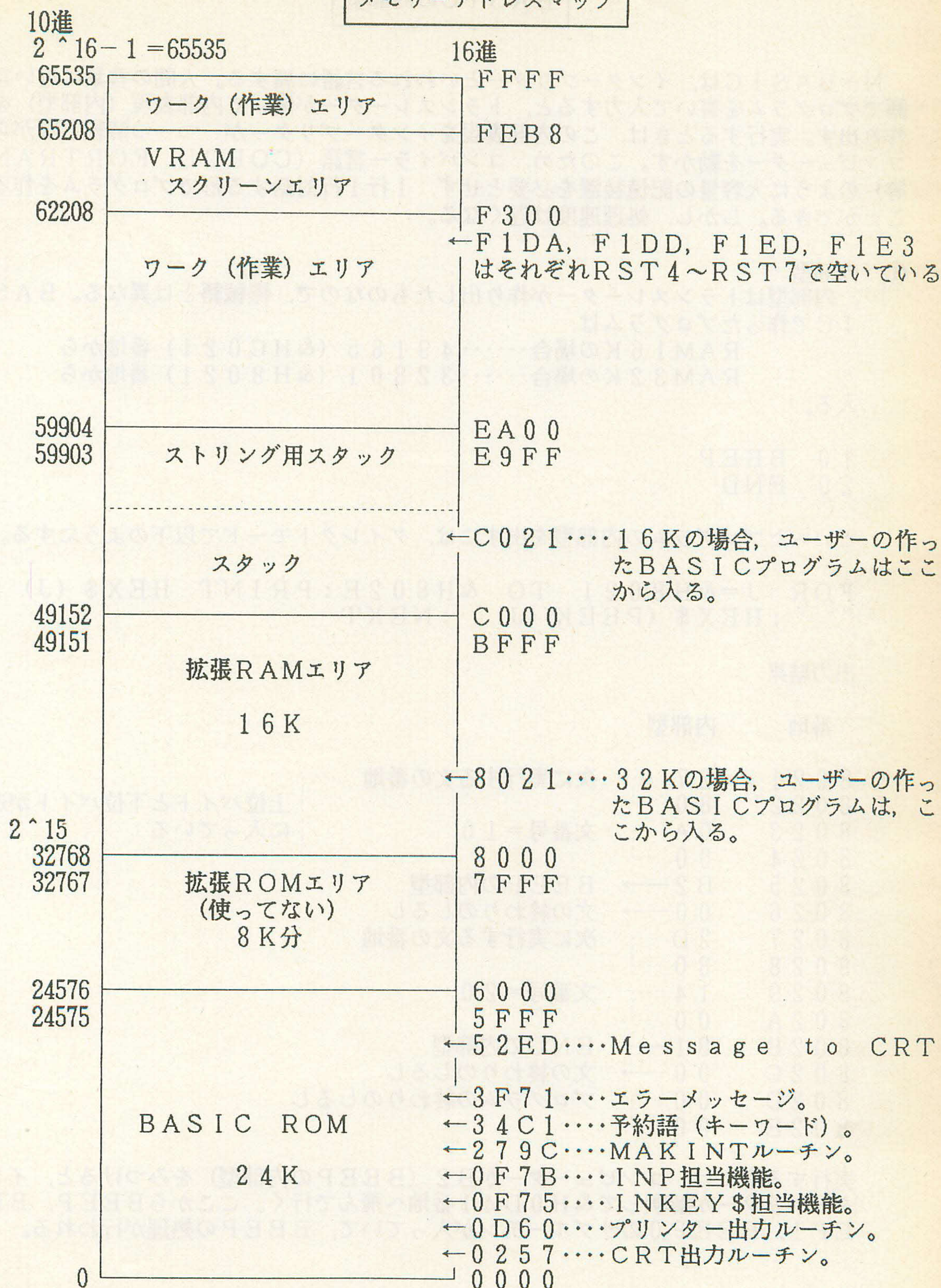
メモリーの種類

ＲＯＭ……………………Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙの略。読み出し専用のメモリー。書き込むことはできない。ＰＣ－８００１の場合は２４ＫバイトのＲＯＭがあり，そこにＮ－ＢＡＳＩＣが入っている。３２Ｋバイトまで拡張できる。

ＰＲＯＭ………………Ｐｒｏｇｒａｍｍａｂｌｅ　Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙの略。読み出し専用のメモリーだが，ＰＲＯＭライターという装置を使えば，プログラムを書き込むことができる。

ＲＡＭ…………………Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙの略。読み書き両方ともできるメモリー。電源を切るとメモリーの内容も消えてしまう。ＰＣ－８００１の場合は，１６Ｋバイトあり，３２Ｋバイトまで拡張できる。

メモリーアドレスマップ



ＢＡＳＩＣの内部型

Ｎ－ＢＡＳＩＣは，インタープリターといわれる言語に属する。人間の言葉に近い言語でプログラムを書いて入力すると，トランスレーターが働いて内部表現（内部型）を作り出す。実行するときは，この内部表現をインタープリターが一つ一つ解釈しながら，コンピューターを動かす。このため，コンパイラー言語（ＣＯＢＯＬ，ＦＯＲＴＲＡＮ等）のように大容量の記憶装置を必要とせず，１行１行対話する形でプログラムを作ることができる。しかし，処理速度は遅くなる。

① 内部型

内部型はトランスレーターが作り出したものなので，機械語とは異なる。ＢＡＳＩＣで作ったプログラムは，

ＲＡＭ１６Ｋの場合……４９１８５（＆ＨＣ０２１）番地から
ＲＡＭ３２Ｋの場合……３２８０１（＆Ｈ８０２１）番地から

入る。

```
10  BEEP
20  END
```

といったプログラムの内部型を出すには，ダイレクトモードで以下のようにする。

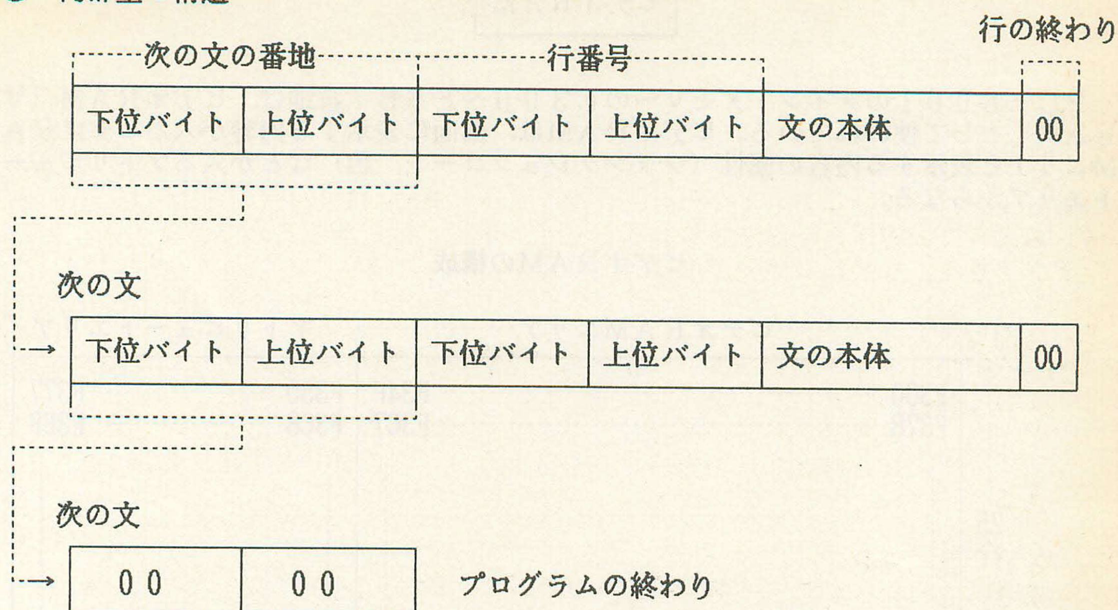
```
FOR J=&H8021 TO &H802E:PRINT HEX$(J);
" ";HEX$(PEEK(J)):NEXT
```

出力結果

番地	内部型	
8021	27	次に実行する文の番地（上位バイトと下位バイトが逆に入っている）
8022	80	
8023	0A	文番号＝１０（上位バイトと下位バイトが逆に入っている）
8024	00	
8025	B2	→ ＢＥＥＰの内部型
8026	00	→ 文の終わりのしるし
8027	2D	次に実行する文の番地
8028	80	
8029	14	文番号＝２０
802A	00	
802B	81	→ ＥＮＤの内部型
802C	00	→ 文の終わりのしるし
802D	00	プログラムの終わりのしるし
802E	00	

実行する際には，コンピューターがＢ２（ＢＥＥＰの内部型）をみつけると，インタープリターが解釈して＆Ｈ０Ｄ４１番地へ飛んで行く。ここからＢＥＥＰ，ＢＥＥＰ１，ＢＥＥＰ０のサブルーチンが入っていて，ＢＥＥＰの処理が行われる。

② 内部型の構造

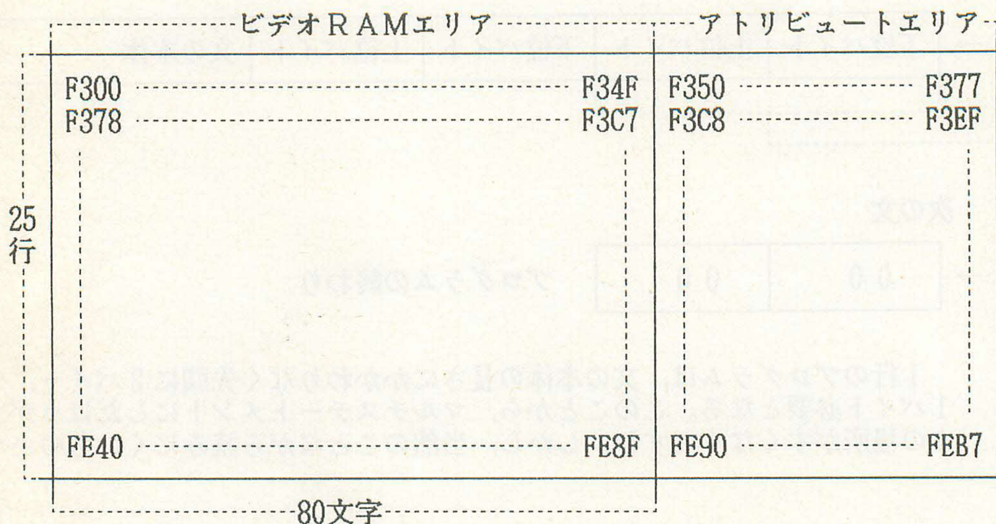


1行のプログラムは，文の本体の長さにかかわりなく先頭に2バイト，末尾に1バイト必要となる。このことから，マルチステートメントにしたほうがプログラムの場所がすくなくてすむ。しかし，当然のことながら読みにくいものとなる。

ビデオＲＡＭ

ＰＣ－８００１のメイン・メモリーのＦ３００～ＦＦＢ７番地は，ビデオＲＡＭ（ＶＲＡＭ）として使われている。ビデオＲＡＭは，画面に表示する内容が入るビデオＲＡＭエリアと表示する内容の属性（ファンクションコード，色）などが入るアトリビュートエリアからなる。

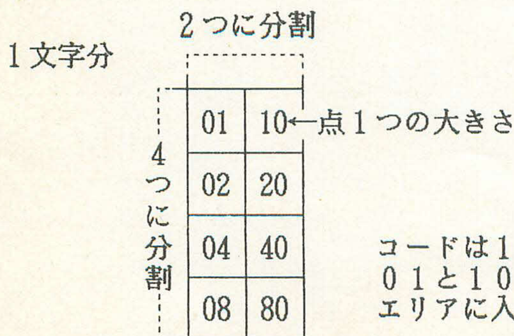
ビデオＲＡＭの構成



① ビデオＲＡＭエリア

画面上に表示される文字コード１バイトが，画面上の位置に対応した場所に入る。４０文字モードの時は，偶数番地が使用され，奇数番地の内容は表示されない。カラーモードの時は，各行の先頭１番地を使用できない。２０行の指定がされた時は，最初から２０行分だけが使用される。

点（ドット）を表示するときは１文字を横２つ，縦４つに分割した大きさが１つの点になる。画面全体に表示できる点の数は，横〔２×（１行の文字数）〕×縦〔４×（行数）〕となる。点が表示されたとき，ビデオＲＡＭエリアの対応する番地には，以下のコードが入る。



コードは１６進数で，点の位置に対応する。
０１と１０の位置に点があれば１１がビデオＲＡＭエリアに入る。

② アトリビュートエリア

ビデオRAMエリアと対応し，画面表示の属性が入っている。属性に変化があるとここに記録される。キャラクターモードで1行に多数の点（ドット）を表示したとき，正しく出ないことがある。これは，基本属性がキャラクターにもかかわらず点（グラフィック）を表示したことにより，アトリビュートエリアが足りなくなってしまったためである。グラフィックモードにすれば解消する。1行に20回以上のアトリビュート変更があると正しく表示されない。

＊アトリビュートエリアの構成

アトリビュートエリアは，次の2バイトを1組として扱われる。

アトリビュート変更点の位置	アトリビュートコード

未使用のとき，アトリビュート変更点の位置には50（16進数）が入る。

＊アトリビュートコードの値（16進数）

ファンクションコード	コンソールモード			
	白黒		カラー	
	キャラクター	グラフィック	キャラクター	グラフィック
0	00	80	08	18
1	01	81	28	38
2	02	82	48	58
3	03	83	68	78
4	04	84	88	98
5	05	85	A8	B8
6	06	86	C8	D8
7	07	87	E8	F8

(例) キャラクターモードでPSET　(0，0) とするとアトリビュートエリアの先頭（F350，F351）は

00	80

となる。

＊ＬＩＮＥ文でファンクションコードを指定した場合

カラーモードで使用したとき，アトリビュートエリアの各行の先頭2バイトは使用されない。ＬＩＮＥ<画面の行>，<ファンクションコード> で行単位のファンクションコードを指定した場合，この場所に以下のようなアトリビュートコード（１６進数）が入る。

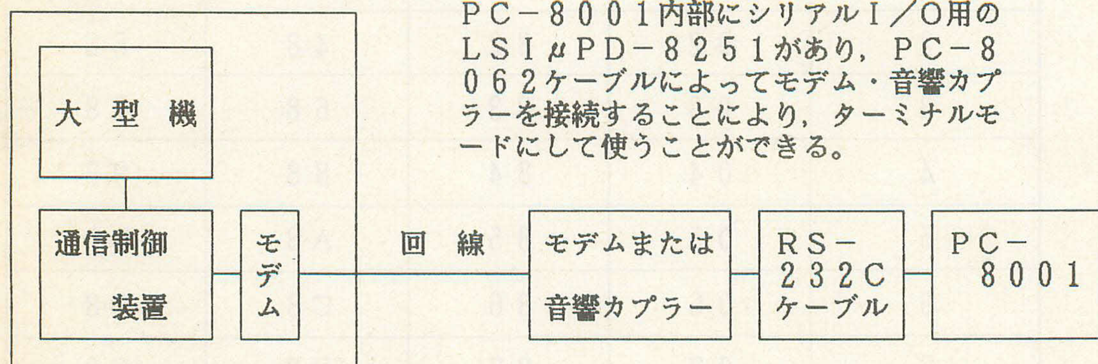
ファンクションコード	アトリビュートコードの値
0	00
1	02
2	04
3	06

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

ターミナルモード

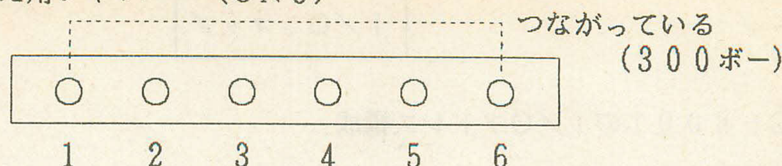
ＰＣ－８００１は，他のコンピューターのターミナルとしても使えるようにターミナルモードを持っている。

ＰＣ－８００１内部にシリアルＩ／Ｏ用のＬＳＩμＰＤ－８２５１があり，ＰＣ－８０６２ケーブルによってモデム・音響カプラーを接続することにより，ターミナルモードにして使うことができる。



計算センターによって通信方法はまちまちであるので，ＴＥＲＭコマンドのパラメーターで調整する。なお，基準周波数はＰＣ内部の基板のジャンパー線を切り替えて設定する。ＣＮ８（ボーレート切り替え用）ジャンパーは３００ボーに設定されている。

＊ボーレート切り替え用ジャンパー（CN8）



ジャンパー	ボーレート	
接続	×16モード	×64モード
1－2	4800	1200
1－3	2400	600
1－4	1200	300
1－5	600	150
1－6	300	75

＊モデム

コンピューターから出力された信号は，通常，直流信号のため，遠方へ伝送すると途中で波形が崩れてしまい正しく伝えることができない。そこで，モデム（変復調装置）によって交流信号に変換して電話回線などを通し伝送し，受け側で再びモデムで直流信号に変換して使う。

＊RS－232Cインタフェース

RS－232CはEIA（米国電子工業会）が決めたコンピューターとモデムとの間のインタフェース（接続手法・装置）規格のこと。RS－232Cインタフェース規格のものであれば，たがいに接続できる。

＊IEEE－488インタフェース

計測器と計測器，コンピューターと計測器を接続する場合の接続インタフェース規格のこと。IEEE－488インタフェースを持った計測器などは，PCファミリーの拡張ユニットPC－8011を通してPC－8001のN－BASICで制御できる。

＊USART

Universal Synchronous／Asynchronous Receiver／Transmitterの略。汎用同期／非同期送受信器。

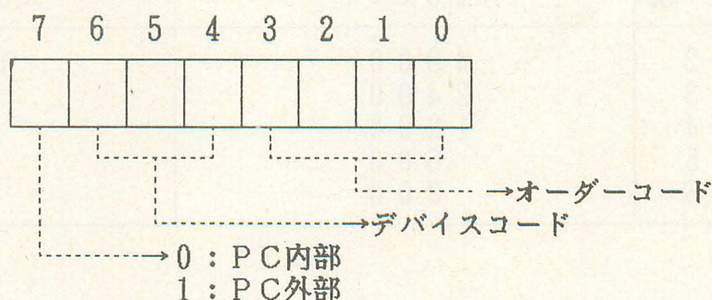
PCの内部では，データを8ビットのパラレル（並列）で扱う。また，コンピューター間のデータ伝送にはシリアル（直列）伝送を使う。USARTはパラレルデータとシリアルデータの仲立ちをして，相互に変換をする機能を持つLSIである。

PC－8001では，μPD8251というLSIが使われていて，モデムの制御，パリティービット（伝送チェック用）の生成やチェックなどを自動的にする。

I／O　マップ

PC－8001のI／Oアドレス構成

I／Oアドレスは，1バイト（8ビット）で表されるので，256の番地（0～255）がある。PCではビット7をPC本体の外部，内部の区別に使い，ビット4，5，6を接続先のデバイス（装置）の区別に使っている。残りのビット0，1，2，3は，それぞれのデバイスにより使われかたがちがう。



内部外部	デバイスコード	アドレス（16進）	インタフェース先	備　考
0 PC内部	000	00～0F	（IN）キーボード00～0A	キーボードマップをみよ
	001	10～1F	（OUT）プリンター，タイマー	
	010	20～2F	（IN，OUT）シリアルチャンネル USART8251	
	011	30～3F	CPUシステムコントロール	
	100	40～4F	（IN）プリンター，BUSY ACK （OUT）プリンター，BEEP	
	101	50～5F	CRTコントロール	
	110	60～6F	DMAコントロール	
	111	70～7F	未使用	

Ｉ／Ｏ　マップ

内部 外部	デバイス コード	アドレス	インタフェース先	備　　考
１ ＰＣ 外部	０００ ００１ ０１０	８０～ＡＦ	拡張用Ｉ／Ｏアドレス ユーザーに開放されている。	
	０１１	Ｂ０～ＢＦ	汎用パラレルＩ／Ｏポート	Ｂ０‥‥８ビット入力ポート　正論理 Ｂ１‥‥８ビット出力ポート　正論理 Ｂ２‥‥４ビット入力ポート　データバスの下位４ビットを読み込む Ｂ３‥‥４ビット出力ポート　負論理　ＰＣ８００１のリセットによりハイレベルになる
	１００	Ｃ０～ＣＦ	ＲＳ２３２Ｃインタフェース	Ｃ１‥‥チャンネル１ Ｃ２‥‥チャンネル２
	１０１	Ｄ０～ＤＦ	ＩＥＥＥ－４８８インタフェース	Ｄ０‥‥データ出力 Ｄ１‥‥データ入力 Ｄ２‥‥制御信号出力 Ｄ３‥‥８２５５コントロール Ｄ８‥‥制御信号入力
	１１０	Ｅ０～ＥＦ	ＰＣ－８０１１システムコントロール	Ｅ０‥‥モード０ Ｅ１‥‥モード１ Ｅ２‥‥モード２ Ｅ３‥‥モード３ Ｅ４‥‥ＲＳ２３２Ｃチャンネル使用初期設定
	１１１	Ｆ０～ＦＦ	フロッピーディスクコントロール	

キーボードマップ

データ・バス

ポート番号	D0	D1	D2	D3	D4	D5	D6	D7
00	0	1	2	3	4	5	6	7
01	8	9	*	+	=	,	.	RETURN
02	@ ゛	A チ	B コ	C ソ	D シ	E イ	F ハ	G キ
03	H ク	I ニ	J マ	K ノ	L リ	M モ	N ミ	O ラ
04	P セ	Q タ	R ス	S ト	T カ	U ナ	V ヒ	W テ
05	X サ	Y ン	Z ッ ツ	「 〔 。	¥ ー	」 〕 ム	^ ヘ	= ー ホ
06	ヲ 0 ワ	! 1 ヌ	" 2 フ	# ァ 3 ア	$ ゥ 4 ウ	% ェ 5 エ	& ォ 6 オ	' ャ 7 ヤ
07	(ュ 8 ユ	) ョ 9 ヨ	* : ケ	+ ; レ	< 、 , ネ	> 。 . ル	? ・ / メ	_ ロ
08	HOME CLR	↓ ↑	← →	INS DEL	GRAPH	カナ	SHIFT	CTRL
09	STOP	f.1	f.2	f.3	f.4	f.5	SPACE	ESC
	254	253	251	247	239	223	191	127

→ データバスの内容，何も押さなければ255 。

キーボードマップ　(グラフィックモード)

データ・バス

ポート番号	D0	D1	D2	D3	D4	D5	D6	D7
00	└	├	┼	┤	╞	╪	╡	┌
01	┬	┐	─	═	│	┴	┘	
02	▌	╰	▬	▬	◥	◢	◤	●
03	○	♠	♦	♣	■	■	■	♥
04	▊	╭	◣	╯	╱	╳	▬	╮
05	▂	╲	＿		円		▌	▐
06	秒					年	月	日
07	時	分	▔	┃	■	▏	▕	
08					GRAPH			
09								
	254	253	251	247	239	223	191	127

（1） 関数

予約語マップ

関数は，頭にＦＦがつき，２バイトでできている。
ＳＱＲの内部コードはＦＦ８７

上段　予約語
下段　サブルーチン入り口の番地

	8	9	A	B	C	D	E	F
0		INP 56A1	CSNG 27B3					
1	LEFT$ 54F9	POS 5079	CDBL 27DF					
2	RIGHT$ 5529	LEN 548C	FIX 282C					
3	MID$ 5532	STR$ 527A	CVI F0E1					
4	SGN 2686	VAL 5553	CVS F0E4					
5	INT 283F	ASC 5498	CVD F0E7					
6	ABS 2671	CHR$ 54A8	DSKF F159					
7	SQR 31A1	PEEK 5911	EOF F0EA					
8	RND 3283	SPACE$ 54DF	LOC F0ED					
9	SIN 32FC	OCT$ 5270	LOF F0F0					
A	LOG 2503	HEX$ 5275	FPOS F168					
B	EXP 31F3	LPOS 5074	MKI$ F0F3					
C	COS 32F6	PORT 20F9	MKS$ F0F6				IEEE	
D	TAN 335D	DEC F123	MKD$ F0F9					
E	ATN 3372	BCD$ F120						
F	FRE 5051	CINT 277F						

(2)　コマンド，ステートメント等　**予約語マップ**

上段　予約語
下段　サブルーチン入り口の番地

	8	9	A	B	C	D	E	F
0		STOP 432A	WIDTH 0843	PRESET 0705	MAT F111	LSET F144	ERR	> (注)
1	END 432F	PRINT 4742	ELSE 45C0	PSET 06B8	LISTEN F10E	RSET F147	STRING$	= (注)
2	FOR 4159	CLEAR 44E8	TRON 4396	BEEP 0D41	DSKO$ F12C	SAVE F14A	USING	< (注)
3	NEXT 4A08	LIST 570C	TROFF 4397	FORMAT F12F	REMOVE F150	LFILES F126	INSTR	+ (注)
4	DATA 45BE	NEW 3DE0	SWAP 439C	KEY 1343	MOUNT F153	INIT 2262	'	− (注)
5	INPUT 48DA	ON 4642	ERASE 43DF	COLOR 0951	OPEN F0FF	LOCATE 0792	VARPTR	* (注)
6	DIM 4E37	WAIT 56B3	ERROR 46C4	TERM 0DB8	FIELD F102		CSRLIN	/ (注)
7	READ 4939	DEF 50CC	RESUME 468C	MON 0D5D	GET 1886	TO	ATTR$	^ (注)
8	LET 45DE	POKE 5918	DELETE 58D9	CMD F0FC	PUT 1891	THEN	DSKI$	AND
9	GOTO 456D	CONT 4383	AUTO 46CF	MOTOR 0DA1	SET F156	TAB(	INKEY$	OR
A	RUN 453D	CSAVE 1EC0	RENUM 5AED	POLL F114	CLOSE F135	STEP	TIME$	XOR
B	IF 4702	CLOAD 1F10	DEFSTR 445B	RBYTE F11A	LOAD F138	USR	DATE$	EQV
C	RESTORE 4302	OUT 56AD	DEFINT 445E	WBYTE F117	MERGE F13B	FN		IMP
D	GOSUB 4555	LPRINT 473A	DEFSNG 4461	ISET F105	FILES F14D	SPC(	SRQ	MOD
E	RETURN 45A3	LLIST 5707	DEFDBL 4464	IRESET F108	NAME F13E	NOT	STATUS	¥ (注)
F	REM 45C0	CONSOLE 0884	LINE 4877	TALK F10B	KILL F141	ERL	POINT	

(注) 演算子としては，この内部コードになる。
ＥＬＳＥは頭に１バイトつき，３ＡＡ１の内部コードとなる。
アポストロフィー（' ）は頭に２バイトつき，３Ａ８ＦＥ４の内部コードとなる。

機械語プログラムの作り方

（1）　モニタープログラムは，ＰＣ内部に記憶されていて，１６進数を使って人間とコンピューターとの仲立ちをしてくれる。
ＰＣ－８００１には8個のモニターコマンドがある。（１６ページ参照）

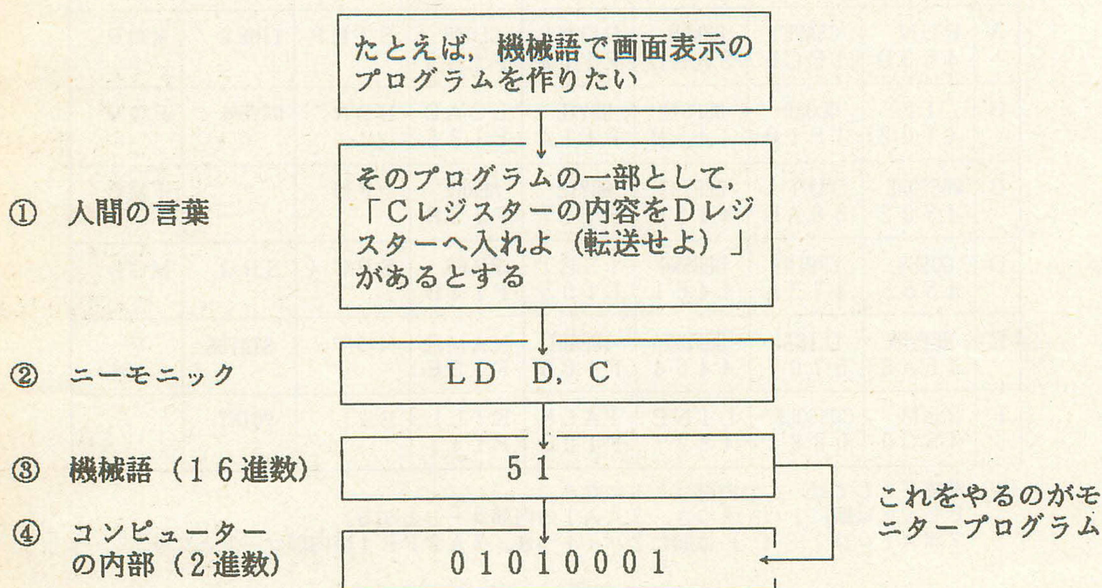
（2）　ＭＯＮ　（ＲＥＴキー）でモニターモードになり，画面左端にアスタリスク（＊）が表示される。このアスタリスクがＢＡＳＩＣモードのときの「ＯＫ」に相当する。
＊の次にＤと入力してみよ。
これでメモリーの００００番地から０００Ｆ番地までの内容が画面に表示される。表示されたＦ３，３１ …… は，Ｎ－ＢＡＳＩＣの中身そのものであり，このように１６進数の機械語で出てくる。

（3）　コンピューターの内部では，電圧の高低を利用し（高いところを１とすれば低いところが０）これを２進法の１と０とにみたてて，すべての作業をしている。
しかし，２進数では書くのがめんどうなので，２進数４桁を１６進数１桁として表示する方法がとられる。１バイト分は次のようになる。

１バイト＝２進数８桁＝１６進数２桁

モニターモードでは，１６進数２桁で入力すれば，モニタープログラムが働いて，コンピューターが理解できる２進数に変換してくれる。

（4）　人間にとっては１６進数でも分かりにくい。そこでニーモニック（Ｍｎｅｍｏｎｉｃ）で表現することが考えられた。ニーモニックには，「記憶する」という意味がある。



(5)　最終的には，④の段階の２進数にしなければならないが，ＰＣでは③の段階の１６進数で入力すれば，モニタープログラムが働いて④の段階に直してくれる。

②のニーモニックの段階で入力して④の段階に直してくれるものをアセンブラーという。したがって，ＰＣ用のアセンブラーを作りたければ，ニーモニック→機械語変換表とそれを索引して変換してくれるプログラムが要る。逆に，機械語をニーモニックに変換してくれるプログラムを逆アセンブラーという。

(6)　ハンドアセンブルとは，アセンブラーの役目を人間が手作業でやることだから，ニーモニックで書いたものを，ニーモニック→機械語変換表をみて１６進数に直すことをいう。

(7)　機械語を入力するには，まず，モニターモードにしておく。

＊Ｓ××××番地（ＲＥＴキー）で××××番地から入力できる。入力した機械語の確認は＊Ｄ××××番地（ＲＥＴキー）で行う。

(8)　機械語プログラムを実行するときも，最初にモニターモードにしておかなければならない。特殊な例としてＤＩＳＫの場合があるが，普通は使わないので省略する。

＊Ｇ××××番地（ＲＥＴキー）で××××番地からの機械語プログラムが動きだす。

(9)　機械語プログラムがＨＡＬＴ（機械語では７６Ｈ）で終わると，機械自体が停止してしまうので，制御できなくなる。リセットボタンを押せばＢＡＳＩＣモードに戻り，機械語のプログラムは残る。

要は，プログラムが終了したらＭＯＮに戻ればいいわけだから（９７ページの予約語ＭＯＮをみると０Ｄ５Ｄ番地。しかし，ここからさらにサブルーチンへとんで結局は５Ｃ６６番地）　ＪＰ　５Ｃ６６Ｈ　とすれば，モニターモードに戻って　＊　が画面表示される。

ＢＡＳＩＣモードに戻りたければ，ＪＰ　０でＲＥＳＥＴボタンを押した状態（コールドスタート）とするか，または　ＪＰ　８でＳＴＯＰキー＋ＲＥＳＥＴボタンを押した状態（ウォームスタート）とすればよい。

(10)　機械語プログラムだけで，技術計算などをやろうとすると，関数やら浮動小数点アキュムレーターやらをすべて自作しなければならない。したがって，ＢＡＳＩＣモードでやったほうがいいだろう。機械語の分野は，むしろ，計算の要らない画面操作などだろう。ゲームなどの画面処理速度は格段にあがる。

(11)　命令によって機械語の長さは違う。１バイトから４バイトまである。

(12)　回路が簡単になるとか速度があがるとか，製造上の理由から，機械語では番地の上位と下位が「さかさま」になっている。

たとえば　ＣＡＬＬ　０Ｅ２６Ｈ　は　ＣＤ　２６　０Ｅ

(13)　現在番地を基準に，ある番地（相対番地）へジャンプする場合，ＪＲ　ｎを使うが，２バイト命令なのでＰＣカウンターが２番地先を指していることに注意。１０Ｈ番地にＪＲがあり，２０Ｈ番地へジャンプするには，
２０Ｈ－１０Ｈ－２Ｈ＝０ＥＨで，ＪＲ　０ＥＨとなる。逆に５Ｈ番地へ行くには，５Ｈ－１０Ｈ－２Ｈ＝－０ＤＨ　これを２の補数で表現するとＦ３Ｈ
ＪＰ　Ｆ３Ｈ（２進数の補数はビットを反転させて１を足す）

＜機械語プログラムの例１＞

＊機能

キーボード上の任意のキーを押すと音が出る。ＳＴＯＰキーを押すとＢＡＳＩＣモードに戻る。

＊ポイント

音を出すのは，ＢＥＥＰの機能を利用している。画面消去をするのは，きれいな音を出すため。（文字コードは７８ページのキャラクターコード表を見よ）

また，ＣＡＬＬした０Ｆ７５Ｈ番地はＮ－ＢＡＳＩＣのルーチンだが，これで押されたキーのキャラクターコードがＡレジスターに入る。

番地	機械語	ニーモニック表示		
Ｄ０００	３Ｅ００	ＬＤ	Ａ，００Ｈ	画面消去
Ｄ００２	Ｄ３５１	ＯＵＴ	（５１Ｈ），Ａ	５１ＨはＩ／Ｏマップ参照
Ｄ００４	ＣＤ７５０Ｆ	ＣＡＬＬ	０Ｆ７５Ｈ	押されたキーがＡレジスターに
Ｄ００７	ＦＥ０３	ＣＰ	０３Ｈ	ＳＴＯＰキーが押されたら
Ｄ００９	２８１６	ＪＲ	Ｚ，１６Ｈ	Ｄ０２１番地へ
Ｄ００Ｂ	５Ｆ	ＬＤ	Ｅ，Ａ	
Ｄ００Ｃ	０Ｅ００	ＬＤ	Ｃ，００Ｈ	繰り返す回数を２５６回にする
Ｄ００Ｅ	４３	ＬＤ	Ｂ，Ｅ	
Ｄ００Ｆ	３Ｅ２０	ＬＤ	Ａ，２０Ｈ	音を出しはじめる
Ｄ０１１	Ｄ３４０	ＯＵＴ	（４０Ｈ），Ａ	４０ＨはＩ／Ｏマップ参照
Ｄ０１３	１０ＦＥ	ＤＪＮＺ	ＦＥＨ	キーの文字コードの値だけ待つ
Ｄ０１５	４３	ＬＤ	Ｂ，Ｅ	
Ｄ０１６	３Ｅ００	ＬＤ	Ａ，００Ｈ	音を止める
Ｄ０１８	Ｄ３４０	ＯＵＴ	（４０Ｈ），Ａ	４０ＨはＩ／Ｏマップ参照
Ｄ０１Ａ	１０ＦＥ	ＤＪＮＺ	ＦＥＨ	キーの文字コードの値だけ待つ
Ｄ０１Ｃ	０Ｄ	ＤＥＣ	Ｃ	繰り返す回数を１減らす
Ｄ０１Ｄ	２０ＥＦ	ＪＲ	ＮＺ，ＥＦＨ	終わりでなければＤ００Ｅ番地
Ｄ０１Ｆ	１８Ｅ３	ＪＲ	Ｅ３Ｈ	終わりならＤ００４番地へ
Ｄ０２１	Ｃ３０８００	ＪＰ	０８Ｈ	ＢＡＳＩＣに戻る（ウォームスタート）

ＢＡＳＩＣモードに戻るのではなく，モニターモードに戻る場合はＤ０２１番地以降Ｄ０５Ｂ番地までに，次の機械語をつけ加えればよい。画面の初期設定をやった後でモニターに戻る。

番地	機械語	番地	機械語	番地	機械語
Ｄ０２１	３Ｅ００	Ｄ０３５	３Ｅ８Ｂ	Ｄ０４９	３Ｅ５３
Ｄ０２３	Ｄ３５１	Ｄ０３７	Ｄ３６５	Ｄ０４Ｂ	Ｄ３５０
Ｄ０２５	３Ｅ８０	Ｄ０３９	３ＥＣＥ	Ｄ０４Ｄ	３Ｅ４３
Ｄ０２７	Ｄ３６８	Ｄ０３Ｂ	Ｄ３５０	Ｄ０４Ｆ	Ｄ３５１
Ｄ０２９	３Ｅ００	Ｄ０３Ｄ	３Ｅ９８	Ｄ０５１	３ＥＣ４
Ｄ０２Ｂ	Ｄ３６４	Ｄ０３Ｆ	Ｄ３５０	Ｄ０５３	Ｄ３６８
Ｄ０２Ｄ	３ＥＦ３	Ｄ０４１	３Ｅ６７	Ｄ０５５	３Ｅ２０
Ｄ０２Ｆ	Ｄ３６４	Ｄ０４３	Ｄ３５０	Ｄ０５７	Ｄ３５１
Ｄ０３１	３ＥＢ７	Ｄ０４５	３ＥＤＥ	Ｄ０５９	Ｃ３６６５Ｃ
Ｄ０３３	Ｄ３６５	Ｄ０４７	Ｄ３５０		

＜機械語プログラムの例２＞

＊機能

キーボードから入力した文字を画面上に出力するとともに，プリンターにも出力する。プリンターは，バッファを持っているのですぐにはプリントしない。ＲＥＴＵＲＮキーを押すか，バッファがいっぱいになるとプリントする。改行をする場合は，ＣＴＲＬキーとＪを押すこと。ＳＴＯＰキーを押すとモニターモードに戻る。

＊ポイント

画面消去・キーボードのどのキーを押したか・画面出力・プリンター出力は，Ｎ－ＢＡＳＩＣ内のルーチンを利用している。

番　地	機械語	ニーモニック表示		
ＣＯＯＯ	ＣＤ５Ａ０４	ＣＡＬＬ	０４５ＡＨ	PRINT CHR $ (12) と同じ
Ｃ００３	ＣＤ７５０Ｆ	ＣＡＬＬ	０Ｆ７５Ｈ	押されたキーがＡレジスターに
Ｃ００６	ＦＥ０３	ＣＰ	０３Ｈ	ＳＴＯＰキーならモニター
Ｃ００８	ＣＡ６６５Ｃ	ＪＰ	Ｚ，５Ｃ６６Ｈ	モードに戻る
Ｃ００Ｂ	ＣＤ２Ｂ００	ＣＡＬＬ	００２ＢＨ	プリンターに出力
Ｃ００Ｅ	ＣＤ３５００	ＣＡＬＬ	００３５Ｈ	画面に出力
Ｃ０１１	１８Ｆ０	ＪＲ	Ｆ０Ｈ	

＜機械語プログラムの例３＞

＊機能

レジスターの内容を画面に表示する。機械語プログラムのテストをするときに便利。テストをするときは，あらかじめＦ１Ｅ３番地から３バイトにＪＰ　Ｅ０００Ｈ（機械語でＣ３００Ｅ０）を入れておく。したがって，レジスターの内容を知りたい場所にＲＥＴ　３８Ｈ（機械語でＦＦ）を入れれば，Ｆ１Ｅ３番地にとび（そこからこのプログラムのＥ０００Ｈ番地へさらにとぶ。Ｆ１Ｅ３番地がフック番地になっている。フックとは「ひっかける」ことで中継番地）このプログラムが実行される。

＊ポイント

５２ＥＤＨ番地の文字列出力ルーチンを利用している。任意の場所でレジスターの内容などを確認するとき，その場所をブレークポイントという。

出力結果

```
ＡＦ　　ＢＣ　　ＤＥ　　ＨＬ
０Ａ４Ａ　５Ｃ５Ｅ　５Ｃ８２　Ｅ０００　　←主レジスターの内容

ＡＦ　　ＢＣ　　ＤＥ　　ＨＬ
２０２４　Ｅ９ＣＦ　０１０１　Ｄ３ＢＡ　　←補助レジスターの内容

ＩＸ　　ＩＹ　　ＰＣ＋１　ＳＰ　　　　←専用レジスター・プログ
９４Ｆ８　９６２Ｅ　４１ＦＡ　Ｅ８Ｄ１　　　ラムカウンター・スタッ
　　　　　　　　　　　　　　　　　　　　ックポインターの内容
```

ＡＦの内容は０Ａ４ＡＨ。したがってＡレジスターの内容は，０ＡＨとなる。０ＡＨは１０進数で１０，２進数で００００１０１０。Ｆレジスターの内容４ＡＨを２進数に直すと，０１００１０１０になる（８３ページＣＰＵレジスター参照）

プログラムの本体は次のようになる。

番地	機械語	ニーモニック表示		
E000	FDE5	PUSH	IY	IX，IYをスタックに退避
E002	DDE5	PUSH	IX	
E004	CD2DE0	CALL	E02DH	主レジスターの表示
E007	08	EX	AF，AF'	レジスター交換
E008	D9	EXX		
E009	CD2DE0	CALL	E02DH	補助レジスターの表示
E00C	CDCA5F	CALL	5FCAH	画面の改行をする
E00F	215BE0	LD	HL，E05BH	IX，IYレジスター等の見出しを表示
E012	CDED52	CALL	52EDH	
E015	CDCA5F	CALL	5FCAH	画面の改行をする
E018	0603	LD	B，03H	
E01A	E1	POP	HL	IX，IY，PC+1の内容を16進4桁で表示
E01B	CDC05E	CALL	5EC0H	
E01E	CDD45F	CALL	5FD4H	
E021	10F7	DJNZ	F7H	
E023	210000	LD	HL，0000H	スタックポインターの値を取り出し16進4桁で表示
E026	39	ADD	HL，SP	
E027	CDC05E	CALL	5EC0H	
E02A	C3665C	JP	5C66H	モニターに戻る
E02D	E5	PUSH	HL	（ここからサブルーチン）HL，DE，BC，AFをスタックに退避
E02E	D5	PUSH	DE	
E02F	C5	PUSH	BC	
E030	F5	PUSH	AF	
E031	CDCA5F	CALL	5FCAH	画面の改行をする
E034	2149E0	LD	HL，E049H	主レジスター，補助レジスターの見出し表示
E037	CDED52	CALL	52EDH	
E03A	CDCA5F	CALL	5FCAH	画面の改行をする
E03D	0604	LD	B，04H	HL，DE，PC，AFの内容を16進4桁で表示
E03F	E1	POP	HL	
E040	CDC05E	CALL	5EC0H	
E043	CDD45F	CALL	5FD4H	
E046	10F7	DJNZ	F7H	
E048	C9	RET		（サブルーチンの終わり）
E049	41462020			主レジスター，補助レジスターの見出し
E04D	20424320			
E051	20204445			
E055	20202048			
E059	4C00			
E05B	49582020			IX，IY，PC+1，SPの見出し
E05F	20495920			
E063	20205043			
E067	2B312053			
E06B	5000			

＜機械語プログラムの例４＞

＊機能

図形を動かす。ドットで描かれた図形が動き続ける。

＊ポイント

８文字分（横４，縦２）の図形をドットに分解してＶＲＡＭ上に描く（Ｃ０６ＥＨ～Ｃ０７５Ｈ番地。８８ページ参照）。図形の点滅はＸＯＲをとることによってする。画面の初期設定はＮ－ＢＡＳＩＣ内のサブルーチンＷＩＤＴＨ（０８４３Ｈ番地。９７ページ予約語マップ参照），ＣＯＮＳＯＬＥ（０８８４Ｈ番地。同），ＣＯＬＯＲ（０９５１Ｈ番地。同）を利用している。いずれもパラメーターの先頭番地をＨＬレジスターに入れて，ＣＡＬＬしてやればよい。Ｃ０７６Ｈ番地以降が各パラメーター。キャラクターコード（７８ページ参照）で入れ，パラメーターの終わりは００Ｈとすること。Ｃ０４ＢＨからのサブルーチンで表示と消去をしている。Ｃ０６４Ｈ番地に画素の先頭番地を入れておき，Ｃ０６６Ｈ～Ｃ０６ＤＨ番地に画素の先頭番地から何番地はなれているかのテーブルを作っておく。先頭番地とテーブルから表示する番地を割り出して，対応する画素をその番地に書き込む。機械語で画面に表示するときは処理速度が速すぎるので適度に時間をかせぐことが必要（Ｃ０４４Ｈがカラまわしのサブルーチン）。Ｃ０２１Ｈ，Ｃ０３３Ｈ番地はＮＯＰで特別な意味はない。ニーモニックは省略する。

番地	機械語	番地	機械語
Ｃ０００	２１７６Ｃ０	Ｃ０４４	Ｃ５
Ｃ００３	ＣＤ４３０８	Ｃ０４５	０６００
Ｃ００６	２１７ＣＣ０	Ｃ０４７	１０ＦＥ
Ｃ００９	ＣＤ８４０８	Ｃ０４９	Ｃ１
Ｃ００Ｃ	２１８５Ｃ０	Ｃ０４Ａ	Ｃ９
Ｃ００Ｆ	ＣＤ５１０９	Ｃ０４Ｂ	０６０８
Ｃ０１２	ＣＤ５Ａ０４	Ｃ０４Ｄ	ＤＤ２１６６Ｃ０
Ｃ０１５	２１００Ｆ３	Ｃ０５１	１６００
Ｃ０１８	２２６４Ｃ０	Ｃ０５３	ＤＤ５Ｅ００
Ｃ０１Ｂ	０６０Ａ	Ｃ０５６	２Ａ６４Ｃ０
Ｃ０１Ｄ	Ｃ５	Ｃ０５９	１９
Ｃ０１Ｅ	ＣＤ４ＢＣ０	Ｃ０５Ａ	ＤＤ７Ｅ０８
Ｃ０２１	００	Ｃ０５Ｄ	ＡＥ
Ｃ０２２	０６００	Ｃ０５Ｅ	７７
Ｃ０２４	ＣＤ４４Ｃ０	Ｃ０５Ｆ	ＤＤ２３
Ｃ０２７	１０ＦＢ	Ｃ０６１	１０ＥＥ
Ｃ０２９	ＣＤ４ＢＣ０	Ｃ０６３	Ｃ９
Ｃ０２Ｃ	０６００	Ｃ０６４	００００
Ｃ０２Ｅ	ＣＤ４４Ｃ０	Ｃ０６６	０００１０２０３
Ｃ０３１	１０ＦＢ	Ｃ０６Ａ	７８７９７Ａ７Ｂ
Ｃ０３３	００	Ｃ０６Ｅ	２１６ＣＣ６１２
Ｃ０３４	２Ａ６４Ｃ０	Ｃ０７２	４８６３３６８４
Ｃ０３７	１６００	Ｃ０７６	３８３０２Ｃ３２
Ｃ０３９	１ＥＦ４	Ｃ０７Ａ	３５００
Ｃ０３Ｂ	１９	Ｃ０７Ｃ	３０２Ｃ３２３５
Ｃ０３Ｃ	２２６４Ｃ０	Ｃ０８０	２Ｃ３０２Ｃ３０
Ｃ０３Ｆ	Ｃ１	Ｃ０８４	００
Ｃ０４０	１０ＤＢ	Ｃ０８５	３０２Ｃ３０２Ｃ
Ｃ０４２	１８Ｄ１	Ｃ０８９	３１００

Ｚ８０相当代表的機械語説明表

語尾のＨは１６進を表す

機械語（１６進）	ニーモニック	説　　明
５１	ＬＤ　　Ｄ，Ｃ	Ｃレジスターの内容をＤレジスターに転送（ロード）せよ ＬＤ　　Ｄ，Ｃ →ソースフィールド →デスティネーションフィールド →オペコードフィールド
０Ｅ　ｎ	ＬＤ　　Ｃ，ｎ	ｎをＣレジスターに転送せよ。ｎは１バイトの数値。16進なら語尾にＨをつける。０Ｅ　７７Ｈなら，７７ＨをＣレジスターに転送せよ。２バイト命令
３６　ｎ	ＬＤ　　（ＨＬ），ｎ	ｎをＨＬレジスターが示す番地に転送せよ。（　）があったら，そのレジスターが示す番地の内容
０１　ｎｎ	ＬＤ　　ＢＣ，ｎｎ	ＬＤ　ＢＣ，４０００Ｈは０１　００Ｈ４０Ｈと上位と下位が逆になることに注意。３バイト命令。なおデータの語尾のＨは，機械語には必要ない
２Ａ　ｎｎ	ＬＤ　　ＨＬ，（ｎｎ）	ＬＤ　ＨＬ，（６０００Ｈ）は２Ａ　００Ｈ６０Ｈ　Ｈに６００１Ｈ番地の内容が，Ｌに６０００Ｈ番地の内容が入る
２２　ｎｎ	ＬＤ　　（ｎｎ），ＨＬ	ＬＤ　（１０００Ｈ），ＨＬは２２　００Ｈ１０Ｈとなる。Ｈの内容を１００１Ｈ番地に転送し，Ｌの内容を１０００Ｈ番地に転送する
Ｃ５	ＰＵＳＨ　ＢＣ	ＢＣの内容をスタックポインターで示す番地へ転送。ＢがＳＰ（スタックポインター）の内容－１番地へ転送され，ＣがＳＰの内容－２番地へ転送される。ＳＰは－２される。あらかじめＳＰにスタックの番地＋１を入れておくこと

語尾のHは１６進を表す

機械語（１６進）	ニーモニック	説　　明
C1	POP　BC	PUSHの逆。SPで示された番地の内容がCに入り，SP＋1で示された番地の内容がBへ入る。SPは＋2される
80	ADD　A，B	A←A＋B　Aレジスターの内容にBレジスターの内容を加え，Aレジスターの内容とする。以下，レジスターの語を省略して説明する
09	ADD　HL，BC	HL←HL＋BC
8B	ADC　A，E	A←A＋E＋CY　CYは桁あふれのキャリーフラグ。演算の結果アキュムレーターの最上位ビットが桁あふれするとキャリーフラグがたつ。2バイト以上の数の和を求めるとき下位バイトどうしの和でキャリーがでたら，この命令で上位バイトどうしの和の方へ加えてやることができる
90	SUB　B	A←A－B
98	SBC　A，B	A←A－B－CY
A0	AND　B	A←A　AND　B
B8	CP　B	A－Bをやり，結果はすててフラグの変化だけを求める。Aは変化しない。比較命令
04	INC　B	B←B＋1　インクリメント命令
05	DEC　B	B←B－1　デクリメント命令
DD86n	ADD　A，(ix＋n)	Aの内容を10H，ixの内容を1000H，nを5Hとすると，1005H番地の内容とAの内容10Hが加算され，Aの内容となる。DDがつくとix（インデックスレジスター）が使われる
FD86n	ADD　A，(iy＋n)	FDがつくとIYレジスター

語尾のHは16進を表す

機械語（16進）	ニーモニック	説明
EB	EX DE, HL	DE←→HL。DEとHLの内容を交換する
D9	EXX	BC, DE, HL←→B' C', D' E', H' L'
E3	EX (SP), HL	H←→（SP+1） L←→（SP）
DDE3	EX (SP), IX	IX（H）←→（SP+1） IX（L）←→（SP）
C3 nn	JP nn	特定番地へジャンプ。JP 3000HはC3 00H30H。3バイト命令
18 n	JR n	現在番地から相対的に指定した番地へとぶ。2バイト命令でプログラムカウンターが2番地先へ進んでいることに注意する。 E000H番地からE012H番地へとびたいときは E012H－E000H－2H＝10H 故に JR 10H
E9	JP (HL)	HLの内容が1234Hなら，そこへとぶ
CA nn	JP Z, nn	Z（ゼロ）フラグを見て，1ならnn番地へとぶ。NZなら0でnnへとぶ
DA nn EA nn E2 nn F2 nn FA nn	JP C, nn	CY（キャリー）フラグをみて1ならnn番地へとぶ 同様にJP PEはパリティー・イーブン（パリティーが偶数なら） 同様にJP POはパリティー・オッド（パリティーが奇数なら） 同様にJP Pはサイン・ポジティブ（正） 同様にJP Mはサイン・ネガティブ（負） 要するに条件が真なら指定番地へとぶ

語尾のHは１６進を表す

機械語（１６進）	ニーモニック	説　　明
１０ ｎ	ＤＪＮＺ ｎ	ＤＥＣ　ＢとＪＲ　ＮＺとの合成。Ｂ←Ｂ－１をやって０でなければジャンプ。０なら次の命令を実行。ジャンプ番地の計算はＪＲ　ｎとおなじ
ＣＤ ｎｎ	ＣＡＬＬ　ｎｎ	ｎｎ番地からはじまるサブルーチンへとぶ
Ｃ４ ｎｎ	ＣＡＬＬ ＮＺ，ｎｎ	ゼロフラグがたっていなければ（ＮＺ），ｎｎ番地へとぶ
Ｃ９	ＲＥＴ	サブルーチンの最後へ書くリターン命令。ＣＡＬＬの次の命令へ帰る
Ｃ８	ＲＥＴ　Ｚ	Ｚ（ゼロ）ならもどる
ＥＤ４Ｄ	ＲＥＴＩ	基本的にはＲＥＴだが，割り込み処理ルーチンに使われ，割り込み終了を周辺装置に知らせる
ＥＤ４５	ＲＥＴＮ	ＮＭＩ（ノンマスカブル割り込み）に対するサービスルーチンの終了
ＤＦ	ＲＳＴ　１８Ｈ	ＲＳＴ（リスタート）はＣＡＬＬと同じ働きをする。この例は１８Ｈを４桁の００１８Ｈ番地とみて，そこがサブルーチン開始番地となる
ＥＤＡ０	ＬＤＩ	（ＨＬ）番地の内容を（ＤＥ）番地の内容に書く。続いてＨＬとＤＥは＋１され，ＢＣは－１される。ブロック転送命令
ＥＤＢ０	ＬＤＩＲ	同上。ＢＣが０になったらブロック転送終了。すなわちＢＣはバイトカウンター。１００バイト転送したければＢＣに１００をセットする
ＥＤＡ８	ＬＤＤ	（ＤＥ）←（ＨＬ）は同じ。ＤＥ，ＨＬ，ＢＣとも－１される
ＥＤＢ８	ＬＤＤＲ	同上。ＢＣが０になったら，ブロック転送を止める

語尾のＨは１６進を表す

機械語（１６進）	ニーモニック	説　　明
ＥＤＡ１	ＣＰＩ	Ａの内容と（ＨＬ）番地の内容とを比較。Ａ，（ＨＬ）は不変。Ｆレジスターのの内容をＺがＡ＝（ＨＬ）なら１，Ｐ／ＶがＢＣ－１≠０なら１。比較後，ＢＣは－１，ＨＬは＋１される
ＥＤＢ１	ＣＰＩＲ	同上。Ａ＝（ＨＬ）かＢＣ＝０のどちらかが成立すると命令実行終了
ＥＤＡ９	ＣＰＤ	ＣＰＩと同じだが，比較後，ＢＣは－１，ＨＬも－１される
ＥＤＢ９	ＣＰＤＲ	ＣＰＩＲと同じ機能。ＢＣ，ＨＬとも－１される点が違う。条件成立までＣＰＤと同じ操作を繰り返す
２７	ＤＡＡ	２進の加減算命令で，ＢＣＤ（２進化１０進）表現の数値間の計算をした場合，ＢＣＤコードに直す ２進計算なら ２７Ｈ＋５８Ｈ＝７ＦＨ ＢＣＤ計算なら ２７Ｈ＋５８Ｈ＝８５Ｈ Ａの内容が７ＦＨとなっているのを８５Ｈに直す。ＡＤＤやＡＤＣ，ＳＵＢ，ＳＢＣと組み合わせて使う 例　ＡＤＤ　Ａ，Ｃ ＤＡＡ
２Ｆ	ＣＰＬ	Ａ←ＮＯＴ　Ａ　Ａの内容の０を１に，１を０にする
ＥＤ４４	ＮＥＧ	Ａ←０－Ａ　Ａの内容を正から負，負から正にする。すなわちＡの内容を２の補数にする
３Ｆ	ＣＣＦ	キャリーフラグが反転する
３７	ＳＣＦ	セットキャリーフラグ。キャリーフラグを１にする
００	ＮＯＰ	ノーオペレーション。なにもしないで次の命令へ行く

語尾のHは１６進を表す

機械語（１６進）	ニーモニック	説　　明
７６	ＨＡＬＴ	ＣＰＵ待機。Ｚ８０のプログラムの実行を停止する
Ｆ３	ＤＩ	ディスエーブル・インタラプト。割り込みを禁止する
ＦＢ	ＥＩ	イネーブル・インタラプト。割り込みを可能にする
ＥＤ４６	ＩＭ　０	割り込みモード０。割り込みを受けつけると，データバスから割り込み命令を入れて，割り込み処理をさせる。すなわち，コールする番地は周辺デバイスからＣＡＬＬやＲＳＴを入れておこなう
ＥＤ５６	ＩＭ　１	同モード１。割り込みを受けつけると，００３８Ｈ番地を無条件にコール。割り込み処理プログラムは００３８Ｈ番地から入れておかねばならない
ＥＤ５Ｅ	ＩＭ　２	同モード２。割り込みを受けつけると自動的にコール命令を実行する 　コールする番地は， 　上位８ビット‥Ｉレジスターから 　下位８ビット‥データバスから ただし，最下位ビットは０として間接コール 　ＩＭ　０～ＩＭ　１は，ＩＮＴ割り込みのやりかたをきめる 　割り込みはＥＩで可能になるが，割り込みが起こるとＤＩの状態になるので，ＥＩを割り込みルーチンのＲＥＴＩの前にいれＥＩ状態に戻す 　ＮＭＩ（ノンマスカブル・インタラプト）の場合は，割り込みを受けつけると無条件に００６６Ｈ番地からはじまるプログラムをコールする 　これは，ＲＥＴＮで戻れる
ＤＢ　ｎ	ＩＮ Ａ，（ｎ）	入力ポートｎを指定してやると，その内容がＡに入る

語尾のHは１６進を表す

機械語（１６進）	ニーモニック	説明
D3　n	OUT　(n)，A	指定した出力ポートに，Aの内容を出力する
ED50	IN　D，(C)	Cレジスターの内容を，入力ポートアドレスとして，その入力ポートからの入力データをDレジスターに入れる
ED41	OUT (C)，B	Cレジスターの内容を，出力ポートアドレスとして，Bレジスターの内容を出力する
EDA2	INI	Cレジスターの内容を入力ポートアドレスとして，入力データを，HLが示す番地へ書く。HLは＋１されBは－１される
EDA3	OUTI	INIの逆。HLは＋１され，Bは－１される すべてのOUT命令で，アドレスバスの下位８ビット（A７～A０）はCの内容，上位８ビット（A１５～A８）にはBの内容が出力される
EDB2	INIR	INI命令をBレジスターの内容だけ（B＝０になるまで）くりかえす
EDB3	OTIR	OUTI命令を同上くりかえす
EDAA	IND	INIとちがう点は，HLが－１される
EDAB	OUTD	OUTIとちがう点は，HLが－１される
EDBA	INDR	INIRではHLは＋１される INDRではHLが－１される
EDBB	OTDR	OTIRではHLは＋１される OTDRではHLが－１される

語尾のＨは１６進を表す

機械語（１６進）	ニーモニック	説　　明
０７	ＲＬＣＡ	左ローテート・サーキュラー・アキュムレーター　１ビット左回転 ＣＹ←７←０ Ａレジスター
１７	ＲＬＡ	左ローテート・アキュムレーター ＣＹを含み１ビット左回転 ＣＹ←７←０ Ａレジスター
０Ｆ	ＲＲＣＡ	右ローテート・サーキュラー・アキュムレーター　１ビット右回転 ７→０→ＣＹ Ａレジスター
１Ｆ	ＲＲＡ	右ローテート・アキュムレーター ＣＹを含み１ビット右回転 ７→０→ＣＹ Ａレジスター ＲＬＣＢ，ＲＲ（ＨＬ），ＲＬＣ（ＩＸ＋ｎ）などは，レジスターが異なったり，番地の内容が対象になるだけで，以上に準ずる
ＣＢ２３	ＳＬＡ　Ｅ	左シフト命令 ＣＹ←７←０←０ Ｅレジスター

語尾のHは16進を表す

機械語（16進）	ニーモニック	説　　明
CB28	SRA　B	右シフト命令 7→0 →CY Bレジスター
CB3A	SRL　D	右シフト命令 0→7→0 →CY Dレジスター
ED6F	RLD	左ローテート・ディジット・アキュムレーター Aの上位4ビット不変 Aレジスター　7 4 3 0 (HL) 番地　7 4 3 0
ED67	RRD	右ローテート・ディジット・アキュムレーター Aの上位4ビット不変 Aレジスター　7 4 3 0 (HL) 番地　7 4 3 0

機械語（16進）	ニーモニック	説　　明
CB67	BIT 4, A	Aレジスターの4ビット目（5桁目になる）が1であればFレジスターのZに0（NO）を書く。0であればZに1（YES）を書く テストビット命令 Aレジスター 765 4 3210 [0] ↓ [1] S Z H PV N C Fレジスター 例　BIT 4, A 　　JP Z, E000H Aレジスターの4ビット目が0ならE000H番地へとぶ
DDCB n5E	BIT 3, (iX+n)	同上，4バイト命令になる。iXを0123Hとし，nを05Hとすると，BIT 3, (iX+05H)でDDCB 05H 5Eとなる iX [0123] +05H=0128H 0128H 7654 3 210 [0] ↓ [1] S Z H PV N C Fレジスター FDがつくとiYになるのは同じ

機械語（16進）	ニーモニック	説明
CBD9	SET 3, C	セットビット。Cレジスターの3ビット目を1にする Cレジスター 7654 3 210 [| | | |1| | |] ↑ 1
CBA0	RES 4, B	リセットビット。Bレジスターの4ビット目を0にする Bレジスター 7654 3 210 [| | |0| | | |] ↑ 0

（注）1. nは1バイトの数
2. nnは2バイトの数。機械語では上位バイト，下位バイトが逆になる
3. （ ）がつくと，（ ）の中の数が番地として使われる
4. iXを含む命令は頭にDDがつく
5. iYを含む命令は頭にFDがつく
6. DDCB，FDCBの4バイト命令は，3バイト目にnが入る
7. （iX＋n）の場合，nは−128（80H）〜127（7FH）
8. AFとA' F'の交換は，EX AF，A' F'とは書かず，EX AF，AF'と書く

機械語→ニーモニック

(00～2B)

機械語	ニーモニック	機械語	ニーモニック
00	NOP	16	LD D, n
01	LD BC, nn	17	RLA
02	LD (BC), A	18	JR n
03	INC BC	19	ADD HL, DE
04	INC B	1A	LD A, (DE)
05	DEC B	1B	DEC DE
06	LD B, n	1C	INC E
07	RLCA	1D	DEC E
08	EX AF, AF'	1E	LD E, n
09	ADD HL, BC	1F	RRA
0A	LD A, (BC)	20	JR NZ, n
0B	DEC BC	21	LD HL, nn
0C	INC C	22	LD (nn), HL
0D	DEC C	23	INC HL
0E	LD C, n	24	INC H
0F	RRCA	25	DEC H
10	DJNZ n	26	LD H, n
11	LD DE, nn	27	DAA
12	LD (DE), A	28	JR Z, n
13	INC DE	29	ADD HL, HL
14	INC D	2A	LD HL, (nn)
15	DEC D	2B	DEC HL

機械語→ニーモニック

(2C～57)

機械語	ニーモニック	機械語	ニーモニック
2C	INC L	42	LD B, D
2D	DEC L	43	LD B, E
2E	LD L, n	44	LD B, H
2F	CPL	45	LD B, L
30	JR NC, n	46	LD B, (HL)
31	LD SP, nn	47	LD B, A
32	LD (nn), A	48	LD C, B
33	INC SP	49	LD C, C
34	INC (HL)	4A	LD C, D
35	DEC (HL)	4B	LD C, E
36	LD (HL), n	4C	LD C, H
37	SCF	4D	LD C, L
38	JR C, n	4E	LD C, (HL)
39	ADD HL, SP	4F	LD C, A
3A	LD A, (nn)	50	LD D, B
3B	DEC SP	51	LD D, C
3C	INC A	52	LD D, D
3D	DEC A	53	LD D, E
3E	LD A, n	54	LD D, H
3F	CCF	55	LD D, L
40	LD B, B	56	LD D, (HL)
41	LD B, C	57	LD D, A

HL←nn

機械語→ニーモニック

(58～83)

機械語	ニーモニック	機械語	ニーモニック
58	LD E, B	6E	LD L, (HL)
59	LD E, C	6F	LD L, A
5A	LD E, D	70	LD (HL), B
5B	LD E, E	71	LD (HL), C
5C	LD E, H	72	LD (HL), D
5D	LD E, L	73	LD (HL), E
5E	LD E, (HL)	74	LD (HL), H
5F	LD E, A	75	LD (HL), L
60	LD H, B	76	HALT
61	LD H, C	77	LD (HL), A
62	LD H, D	78	LD A, B
63	LD H, E	79	LD A, C
64	LD H, H	7A	LD A, D
65	LD H, L	7B	LD A, E
66	LD H, (HL)	7C	LD A, H
67	LD H, A	7D	LD A, L
68	LD L, B	7E	LD A, (HL)
69	LD L, C	7F	LD A, A
6A	LD L, D	80	ADD A, B
6B	LD L, E	81	ADD A, C
6C	LD L, H	82	ADD A, D
6D	LD L, L	83	ADD A, E

機械語→ニーモニック

(84～AF)

機械語	ニーモニック	機械語	ニーモニック
84	ADD A, H	9A	SBC A, D
85	ADD A, L	9B	SBC A, E
86	ADD A, (HL)	9C	SBC A, H
87	ADD A, A	9D	SBC A, L
88	ADC A, B	9E	SBC A, (HL)
89	ADC A, C	9F	SBC A, A
8A	ADC A, D	A0	AND B
8B	ADC A, E	A1	AND C
8C	ADC A, H	A2	AND D
8D	ADC A, L	A3	AND E
8E	ADC A, (HL)	A4	AND H
8F	ADC A, A	A5	AND L
90	SUB B	A6	AND (HL)
91	SUB C	A7	AND A
92	SUB D	A8	XOR B
93	SUB E	A9	XOR C
94	SUB H	AA	XOR D
95	SUB L	AB	XOR E
96	SUB (HL)	AC	XOR H
97	SUB A	AD	XOR L
98	SBC A, B	AE	XOR (HL)
99	SBC A, C	AF	XOR A

機械語→ニーモニック

(B0～DB)

機械語	ニーモニック	機械語	ニーモニック
B0	OR B	C6	ADD A, n
B1	OR C	C7	RST 0H
B2	OR D	C8	RET Z
B3	OR E	C9	RET
B4	OR H	CA	JP Z, nn
B5	OR L	CB	121～126ページ参照
B6	OR (HL)	CC	CALL Z, nn
B7	OR A	CD	CALL nn
B8	CP B	CE	ADC A, n
B9	CP C	CF	RST 8H
BA	CP D	D0	RET NC
BB	CP E	D1	POP DE
BC	CP H	D2	JP NC, nn
BD	CP L	D3	OUT (n), A
BE	CP (HL)	D4	CALL NC, nn
BF	CP A	D5	PUSH DE
C0	RET NZ	D6	SUB n
C1	POP BC	D7	RST 10H
C2	JP NZ, nn	D8	RET C
C3	JP nn	D9	EXX
C4	CALL NZ, nn	DA	JP C, nn
C5	PUSH BC	DB	IN A, (n)

機械語→ニーモニック

(DC～FF)

機械語	ニーモニック	機械語	ニーモニック
DC	CALL C, nn	F2	JP P, nn
DD	127ページ参照	F3	DI
DE	SBC A, n	F4	CALL P, nn
DF	RST 18H	F5	PUSH AF
E0	RET PO	F6	OR n
E1	POP HL	F7	RST 30H
E2	JP PO, nn	F8	RET M
E3	EX (SP), HL	F9	LD SP, HL
E4	CALL PO, nn	FA	JP M, nn
E5	PUSH HL	FB	EI
E6	AND n	FC	CALL M, nn
E7	RST 20H	FD	127ページ欄外参照
E8	RET PE	FE	CP n
E9	JP (HL)	FF	RST 38H
EA	JP PE, nn		
EB	EX DE, HL		
EC	CALL PE, nn		
ED	128～129ページ参照		
EE	XOR n		
EF	RST 28H		
F0	RET P		
F1	POP AF		

機械語→ニーモニック

(CB00～CB2B)

機械語	ニーモニック	機械語	ニーモニック
CB00	RLC B	CB16	RL (HL)
CB01	RLC C	CB17	RL A
CB02	RLC D	CB18	RR B
CB03	RLC E	CB19	RR C
CB04	RLC H	CB1A	RR D
CB05	RLC L	CB1B	RR E
CB06	RLC (HL)	CB1C	RR H
CB07	RLC A	CB1D	RR L
CB08	RRC B	CB1E	RR (HL)
CB09	RRC C	CB1F	RR A
CB0A	RRC D	CB20	SLA B
CB0B	RRC E	CB21	SLA C
CB0C	RRC H	CB22	SLA D
CB0D	RRC L	CB23	SLA E
CB0E	RRC (HL)	CB24	SLA H
CB0F	RRC A	CB25	SLA L
CB10	RL B	CB26	SLA (HL)
CB11	RL C	CB27	SLA A
CB12	RL D	CB28	SRA B
CB13	RL E	CB29	SRA C
CB14	RL H	CB2A	SRA D
CB15	RL L	CB2B	SRA E

機械語→ニーモニック

(CB2C〜CB5F)

機械語	ニーモニック	機械語	ニーモニック
CB2C	SRA H	CB4A	BIT 1, D
CB2D	SRA L	CB4B	BIT 1, E
CB2E	SRA (HL)	CB4C	BIT 1, H
CB2F	SRA A	CB4D	BIT 1, L
CB38	SRL B	CB4E	BIT 1, (HL)
CB39	SRL C	CB4F	BIT 1, A
CB3A	SRL D	CB50	BIT 2, B
CB3B	SRL E	CB51	BIT 2, C
CB3C	SRL H	CB52	BIT 2, D
CB3D	SRL L	CB53	BIT 2, E
CB3E	SRL (HL)	CB54	BIT 2, H
CB3F	SRL A	CB55	BIT 2, L
CB40	BIT 0, B	CB56	BIT 2, (HL)
CB41	BIT 0, C	CB57	BIT 2, A
CB42	BIT 0, D	CB58	BIT 3, B
CB43	BIT 0, E	CB59	BIT 3, C
CB44	BIT 0, H	CB5A	BIT 3, D
CB45	BIT 0, L	CB5B	BIT 3, E
CB46	BIT 0, (HL)	CB5C	BIT 3, H
CB47	BIT 0, A	CB5D	BIT 3, L
CB48	BIT 1, B	CB5E	BIT 3, (HL)
CB49	BIT 1, C	CB5F	BIT 3, A

機械語→ニーモニック

(CB60〜CB8B)

機械語	ニーモニック	機械語	ニーモニック
CB60	BIT 4, B	CB76	BIT 6, (HL)
CB61	BIT 4, C	CB77	BIT 6, A
CB62	BIT 4, D	CB78	BIT 7, B
CB63	BIT 4, E	CB79	BIT 7, C
CB64	BIT 4, H	CB7A	BIT 7, D
CB65	BIT 4, L	CB7B	BIT 7, E
CB66	BIT 4, (HL)	CB7C	BIT 7, H
CB67	BIT 4, A	CB7D	BIT 7, L
CB68	BIT 5, B	CB7E	BIT 7, (HL)
CB69	BIT 5, C	CB7F	BIT 7, A
CB6A	BIT 5, D	CB80	RES 0, B
CB6B	BIT 5, E	CB81	RES 0, C
CB6C	BIT 5, H	CB82	RES 0, D
CB6D	BIT 5, L	CB83	RES 0, E
CB6E	BIT 5, (HL)	CB84	RES 0, H
CB6F	BIT 5, A	CB85	RES 0, L
CB70	BIT 6, B	CB66	RES 0, (HL)
CB71	BIT 6, C	CB87	RES 0, A
CB72	BIT 6, D	CB88	RES 1, B
CB73	BIT 6, E	CB89	RES 1, C
CB74	BIT 6, H	CB8A	RES 1, D
CB75	BIT 6, L	CB8B	RES 1, E

機械語→ニーモニック

(CB8C~CBB7)

機械語	ニーモニック	機械語	ニーモニック
CB8C	RES 1, H	CBA2	RES 4, D
CB8D	RES 1, L	CBA3	RES 4, E
CB8E	RES 1, (HL)	CBA4	RES 4, H
CB8F	RES 1, A	CBA5	RES 4, L
CB90	RES 2, B	CBA6	RES 4, (HL)
CB91	RES 2, C	CBA7	RES 4, A
CB92	RES 2, D	CBA8	RES 5, B
CB93	RES 2, E	CBA9	RES 5, C
CB94	RES 2, H	CBAA	RES 5, D
CB95	RES 2, L	CBAB	RES 5, E
CB96	RES 2, (HL)	CBAC	RES 5, H
CB97	RES 2, A	CBAD	RES 5, L
CB98	RES 3, B	CBAE	RES 5, (HL)
CB99	RES 3, C	CBAF	RES 5, A
CB9A	RES 3, D	CBB0	RES 6, B
CB9B	RES 3, E	CBB1	RES 6, C
CB9C	RES 3, H	CBB2	RES 6, D
CB9D	RES 3, L	CBB3	RES 6, E
CB9E	RES 3, (HL)	CBB4	RES 6, H
CB9F	RES 3, A	CBB5	RES 6, L
CBA0	RES 4, B	CBB6	RES 6, (HL)
CBA1	RES 4, C	CBB7	RES 6, A

機械語→ニーモニック

(CBB8～CBE3)

機械語	ニーモニック	機械語	ニーモニック
CBB8	RES 7, B	CBCE	SET 1, (HL)
CBB9	RES 7, C	CBCF	SET 1, A
CBBA	RES 7, D	CBD0	SET 2, B
CBBB	RES 7, E	CBD1	SET 2, C
CBBC	RES 7, H	CBD2	SET 2, D
CBBD	RES 7, L	CBD3	SET 2, E
CBBE	RES 7, (HL)	CBD4	SET 2, H
CBBF	RES 7, A	CBD5	SET 2, L
CBC0	SET 0, B	CBD6	SET 2, (HL)
CBC1	SET 0, C	CBD7	SET 2, A
CBC2	SET 0, D	CBD8	SET 3, B
CBC3	SET 0, E	CBD9	SET 3, C
CBC4	SET 0, H	CBDA	SET 3, D
CBC5	SET 0, L	CBDB	SET 3, E
CBC6	SET 0, (HL)	CBDC	SET 3, H
CBC7	SET 0, A	CBDD	SET 3, L
CBC8	SET 1, B	CBDE	SET 3, (HL)
CBC9	SET 1, C	CBDF	SET 3, A
CBCA	SET 1, D	CBE0	SET 4, B
CBCB	SET 1, E	CBE1	SET 4, C
CBCC	SET 1, H	CBE2	SET 4, D
CBCD	SET 1, L	CBE3	SET 4, E

機械語→ニーモニック

(CBE4～CBFF)

機械語	ニーモニック	機械語	ニーモニック
CBE4	SET 4, H	CBFA	SET 7, D
CBE5	SET 4, L	CBFB	SET 7, E
CBE6	SET 4, (HL)	CBFC	SET 7, H
CBE7	SET 4, A	CBFD	SET 7, L
CBE8	SET 5, B	CBFE	SET 7, (HL)
CBE9	SET 5, C	CBFF	SET 7, A
CBEA	SET 5, D		
CBEB	SET 5, E		
CBEC	SET 5, H		
CBED	SET 5, L		
CBEE	SET 5, (HL)		
CBEF	SET 5, A		
CBF0	SET 6, B		
CBF1	SET 6, C		
CBF2	SET 6, D		
CBF3	SET 6, E		
CBF4	SET 6, H		
CBF5	SET 6, L		
CBF6	SET 6, (HL)		
CBF7	SET 6, A		
CBF8	SET 7, B		
CBF9	SET 7, C		

DDXXをFDXXにすると
iXがiYになる。

機械語→ニーモニック

インデックスレジスター関係
(DD09〜DDF9)

機械語	ニーモニック	機械語	ニーモニック
DD09	ADD iX, BC	DD74	LD (iX+n), H
DD19	ADD iX, DE	DD75	LD (iX+n), L
DD21	LD iX, nn	DD77	LD (iX+n), A
DD22	LD (nn), iX	DD7E	LD A, (iX+n)
DD23	INC iX	DD86	ADD A, (iX+n)
DD29	ADD iX, iX	DD8E	ADC A, (iX+n)
DD2A	LD iX, (nn)	DD96	SUB (iX+n)
DD2B	DEC iX	DD9E	SBC A, (iX+n)
DD34	INC (iX+n)	DDA6	AND (iX+n)
DD35	DEC (iX+n)	DDAE	XOR (iX+n)
DD36	LD (iX+n), n	DDB6	OR (iX+n)
DD39	ADD iX, SP	DDBE	CP (iX+n)
DD46	LD B, (iX+n)	DDCB	130ページ参照
DD4E	LD C, (iX+n)	DDE1	POP iX
DD56	LD D, (iX+n)	DDE3	EX (SP), iX
DD5E	LD E, (iX+n)	DDE5	PUSH iX
DD66	LD H, (iX+n)	DDE9	JP (iX)
DD6E	LD L, (iX+n)	DDF9	LD SP, iX
DD70	LD (iX+n), B		
DD71	LD (iX+n), C		
DD72	LD (iX+n), D		
DD73	LD (iX+n), E		

機械語→ニーモニック

(ED40~EDA1)

機械語	ニーモニック	機械語	ニーモニック
ED40	IN B, (C)	ED5A	ADC HL, DE
ED41	OUT (C), B	ED5B	LD DE, (nn)
ED42	SBC HL, BC	ED5E	IM 2
ED43	LDA (nn), BC	ED5F	LD A, R
ED44	NEG	ED60	IN H, (C)
ED45	RETN	ED61	OUT (C), H
ED46	IM 0	ED62	SBC HL, HL
ED47	LD I, A	ED63	LD (nn), HL
ED48	IN C, (C)	ED67	RRD
ED49	OUT (C), C	ED68	IN L, (C)
ED4A	ADC HL, BC	ED69	OUT (C), L
ED4B	LD BC, (nn)	ED6A	ADC HL, HL
ED4D	RETI	ED6B	LD HL, (nn)
ED4F	LD R, A	ED6F	RLD
ED50	IN D, (C)	ED72	SBC HL, SP
ED51	OUT (C), D	ED73	LD (nn), SP
ED52	SBC HL, DE	ED78	IN A, (C)
ED53	LD (nn), DE	ED79	OUT (C), A
ED56	IM 1	ED7A	ADC HL, SP
ED57	LD A, I	ED7B	LD SP, (nn)
ED58	IN E, (C)	EDA0	LDI
ED59	OUT (C), E	EDA1	CPI

機械語→ニーモニック

(EDA2～EDBB)

機械語	ニーモニック	機械語	ニーモニック
EDA2	INI		
EDA3	OUTI		
EDA8	LDD		
EDA9	CPD		
EDAA	IND		
EDAB	OUTD		
EDB0	LDIR		
EDB1	CPIR		
EDB2	INIR		
EDB3	OTIR		
EDB8	LDDR		
EDB9	CPDR		
EDBA	INDR		
EDBB	OTDR		

機械語→ニーモニック　インデックスレジスター関係
(DDCBXX06〜DDCBXXFE)

DDCBXXをFDCBXXとするとiXがiYになる。XX＝n

機械語	ニーモニック	機械語	ニーモニック
DDCBXX06	RLC　(iX+n)	DDCBXXB6	RES　6，(iX+n)
DDCBXX0E	RRC　(iX+n)	DDCBXXBE	RES　7，(iX+n)
DDCBXX16	RL　(iX+n)	DDCBXXC6	SET　0，(iX+n)
DDCBXX1E	RR　(iX+n)	DDCBXXCE	SET　1，(iX+n)
DDCBXX26	SLA　(iX+n)	DDCBXXD6	SET　2，(iX+n)
DDCBXX2E	SRA　(iX+n)	DDCBXXDE	SET　3，(iX+n)
DDCBXX3E	SRL　(iX+n)	DDCBXXE6	SET　4，(iX+n)
DDCBXX46	BIT　0，(iX+n)	DDCBXXEE	SET　5，(iX+n)
DDCBXX4E	BIT　1，(iX+n)	DDCBXXF6	SET　6，(iX+n)
DDCBXX56	BIT　2，(iX+n)	DDCBXXFE	SET　7，(iX+n)
DDCBXX5E	BIT　3，(iX+n)		
DDCBXX66	BIT　4，(iX+n)		
DDCBXX6E	BIT　5，(iX+n)		
DDCBXX76	BIT　6，(iX+n)		
DDCBXX7E	BIT　7，(iX+n)		
DDCBXX86	RES　0，(iX+n)		
DDCBXX8E	RES　1，(iX+n)		
DDCBXX96	RES　2，(iX+n)		
DDCBXX9E	RES　3，(iX+n)		
DDCBXXA6	RES　4，(iX+n)		
DDCBXXAE	RES　5，(iX+n)		

		A	B	C	D	E	H	L
8ビット転送・演算	LD A	7F	78	79	7A	7B	7C	7D
	LD B	47	40	41	42	43	44	45
	LD C	4F	48	49	4A	4B	4C	4D
	LD D	57	50	51	52	53	54	55
	LD E	5F	58	59	5A	5B	5C	5D
	LD H	67	60	61	62	63	64	65
	LD L	6F	68	69	6A	6B	6C	6D
	LD (HL)	77	70	71	72	73	74	75
	LD (BC)	02						
	LD (DE)	12						
	LD (nn)	32						
	LD I	ED47						
	LD R	ED4F						
	ADD	87	80	81	82	83	84	85
	ADC	8F	88	89	8A	8B	8C	8D
	SUB	97	90	91	92	93	94	95
	SBC	9F	98	99	9A	9B	9C	9D
	AND	A7	A0	A1	A2	A3	A4	A5
	XOR	AF	A8	A9	AA	AB	AC	AD
	OR	B7	B0	B1	B2	B3	B4	B5
	CP	BF	B8	B9	BA	BB	BC	BD
	INC	3C	04	0C	14	1C	24	2C
	DEC	3D	05	0D	15	1D	25	2D

ニーモニック→機械語

＊インデックスレジスター関係は１２７ページ，１３０ページを見よ

		(HL)	n	(BC)	(DE)	(nn)	I	R
8ビット転送・演算	LD A	7E	3E	0A	1A	3A	ED57	ED5F
	LD B	46	06					
	LD C	4E	0E					
	LD D	56	16					
	LD E	5E	1E					
	LD H	66	26					
	LD L	6E	2E					
	LD (HL)		36					
	LD (BC)							
	LD (DE)							
	LD (nn)							
	LD I							
	LD R							
	ADD	86	C6					
	ADC	8E	CE					
	SUB	96	D6					
	SBC	9E	DE					
	AND	A6	E6					
	XOR	AE	EE					
	OR	B6	F6					
	CP	BE	FE					
	INC	34						
	DEC	35						

＊インデックスレジスター関係は127ページ，130ページを見よ

ニーモニック→機械語

		AF	BC	DE	HL	SP	nn	(nn)
16ビット転送・演算	PUSH	F5	C5	D5	E5			
	POP	F1	C1	D1	E1			
	LD BC						01	ED4B
	LD DE						11	ED5B
	LD HL						21	2A
	LD SP				F9		31	ED7B
	LD (nn)		ED43	ED53	22	ED73		
	ADD HL		09	19	29	39		
	ADC HL		ED4A	ED5A	ED6A	ED7A		
	SBC HL		ED42	ED52	ED62	ED72		
	INC		03	13	23	33		
	DEC		0B	1B	2B	3B		

注　LD　HL，(nn)は2A。しかし，ED6Bでも使える。また，
LD　(nn)，HLの22はED63でも使える。
インテルの8080（Z80のもとになったCPU）の命令は2A，
22なので，こちらを使った方が無難かもしれない。

入出力	IN A, (n)	DB
	OUT (n), A	D3
	IN A, (C)	ED78
	IN B, (C)	ED40
	IN C, (C)	ED48
	IN D, (C)	ED50
	IN E, (C)	ED58
	IN H, (C)	ED60
	IN L, (C)	ED68
	OUT (C), A	ED79
	OUT (C), B	ED41
	OUT (C), C	ED49
	OUT (C), D	ED51
	OUT (C), E	ED59
	OUT (C), H	ED61
	OUT (C), L	ED69

ブロック入出力・転送・検索	INI	EDA2
	INIR	EDB2
	IND	EDAA
	INDR	EDBA
	OUTI	EDA3
	OTIR	EDB3
	OUTD	EDAB
	OTDR	EDBB
	LDI	EDA0
	LDIR	EDB0
	LDD	EDA8
	LDDR	EDB8
	CPI	EDA1
	CPIR	EDB1
	CPD	EDA9
	CPDR	EDB9

ニーモニック→機械語

ジャンプ	JR n	18
	JR Z, n	28
	JR NZ, n	20
	JR C, n	38
	JR NC, n	30
	JP nn	C3
	JP Z, nn	CA
	JP NZ, nn	C2
	JP C, nn	DA
	JP NC, nn	D2
	JP PE, nn	EA
	JP PO, nn	E2
	JP P, nn	F2
	JP M, nn	FA
コール	CALL nn	CD
	CALL Z, nn	CC
	CALL NZ, nn	C4
	CALL C, nn	DC
	CALL NC, nn	D4
	CALL PE, nn	EC
	CALL PO, nn	E4
	CALL P, nn	F4
	CALL M, nn	FC

リターン	RET	C9
	RET Z	C8
	RET NZ	C0
	RET C	D8
	RET NC	D0
	RET PE	E8
	RET PO	E0
	RET P	F0
	RET M	F8
	RETI	ED4D
	RETN	ED45
リスタート	RST 0H	C7
	RST 8H	CF
	RST 10H	D7
	RST 18H	DF
	RST 20H	E7
	RST 28H	EF
	RST 30H	F7
	RST 38H	FF
交換	EX DE, HL	EB
	EX AF, AF'	08
	EXX	D9
	EX (SP), HL	E3

ニーモニック↓機械語

操作・特殊JP	DAA	27
	CPL	2F
	NEG	ED44
	CCF	3F
	SCF	37
	JP (HL)	E9
	DJNZ n	10
CPUコントロール	EI	FB
	DI	F3
	IM 0	ED46
	IM 1	ED56
	IM 2	ED5E
	NOP	00
	HALT	76
回転	RLD	ED6F
	RRD	ED67
	RLCA	07
	RRCA	0F
	RLA	17
	RRA	1F

週１回２時間　合計１５回でのカリキュラム例

1	スイッチの入れかた　キーボードの説明　ＰＣテスト法（FRE BEEP　TIME＄）ダイレクトモード（PRINT 四則演算　コンマとセミコロン）プログラムモード（INPUT　LET　END）ＲＥＴＵＲＮとＮＥＷの徹底　電源を一度切ったら５秒以上たたなければ再度スイッチを入れてはいけないこととＲＥＳＥＴを押すと初期状態にもどることの徹底　キャラクターモードのＬＩＮＥでの線引き
2	復習　ファンクションキーの再徹底　プログラムの修正方法　円の面積（GOTO　IF～THEN　STOP）フローチャートの説明　キャラクターモードのＬＩＮＥで箱を描く　次回なにをやるか予告すること
3	復習　５個のデータの平均など平均を求めるプログラム　ＦＯＲ～ＮＥＸＴによる繰り返し処理　ＰＳＥＴを使った線引き　Ｘ軸Ｙ軸を引きＰＳＥＴで任意の位置に点を打つ　初心者の第１難関はＦＯＲ～ＮＥＸＴ
4	復習　ＦＯＲ～ＮＥＸＴはフローチャートを描いて再徹底　九九の掛け算で多重ループの説明　ＲＥＡＤ　ＤＡＴＡ　ＴＡＢによる二次方程式　ＦＯＲ～ＮＥＸＴとＰＳＥＴ　ＰＲＥＳＥＴを組み合わせて線　ドットが進む
5	ＲＥＡＤ　ＤＡＴＡの復習　ＧＯＳＵＢとＲＥＴＵＲＮ　ＲＥＭとアポストロフィでの注釈行の入れ方　ＲＥＳＴＯＲＥ　図形（STRING＄を使ってみる）
6	復習　変数の説明（単精度　倍精度　整数　文字変数などをくわしく）配列（DIM）図形はＫの長さの線が進む　出席者が減りだす　抜けた人はついてくることができなくなる　そのひとにはキーボード操作の習熟を
7	復習　二重添字つき変数　図形は一歩後退二歩前進など複雑な動きをするもの　配列に時間をかけること　このあたりからキーボード操作が速くなり　いい質問が出はじめる　さもないと講義は形式的に進行していることになる
8	復習　作表（PRINTUSING）　ＲＮＤとＩＮＴ　ＰＲＩＮＴＵＳＩＮＧでレイアウトすることを十分理解させる
9	復習　情報検索（INSTR をていねいに説明する　番号から名前を求める　その逆　文字検索の意味を理解させる）　出席者はまったく固定する　図形はランダムに箱を描く，箱の中で玉が動くなどを適当に
10	復習　分類（データがどう比較されて入れかわるか経過を出力すると理解が早い　黒板に書いてもよい　文字ソートもできること　SWAPも使ってみる）
11	復習　関数（画面との対話が主だったのでフローチャートからプログラムを書かせてみると３０分はかかる）　図形は円　サインカーブ
12	復習　関数の自由定義（借金返済　元利合計など具体的なものを）乱数の作りかたの例　図形をやる時間はないだろう

13	復習　２進１０進１６進　キャラクターコード表の説明　文字から数値へ数値から文字への変換　プログラムをとりおく（CLOAD CSAVE など　デモ用カセットを用意する）
14	復習　折れ線グラフなど（受講者のテーマに合わせる）
15	復習　ヒストグラムなど（受講者のテーマに合わせる）

＊講習の注意点

（1）２時間を１時間３０分と３０分に分ける。後半の３０分を図形にあてる。
（2）１つのポイントは『キーボードをいかに速く習熟させるか』にある。カナ文字を使ったプログラムは講義を遅らせる。あらかじめ，キーボード図を渡しておく。大きなキーボード図も掲示用につくっておく。
（3）ファンクションキーを最初に教えておくと入力がかなり楽になる。
特に，ＲＥＴＵＲＮキーは，必ず押すような癖をつけさせる。
（4）常識外のエラーはすべてキーボードの誤操作にある。
（5）予習復習をさせないと，週１回の講義では忘れてしまう。次回なにをやるか予告すること。
（6）プリンターがあれば，あらかじめ当日のプログラム・リストを受講生分だけ作ってやるくらいの親切さは欲しい。ノートをとることに熱中する人がいる。
（7）プログラムはあらかじめカセットテープにとっておいて，それを使う手もある。
（8）１５回では，機械語に連結していくＵＳＲ関数までやることはできない。
（9）他クラスのひとが振り替え受講するような場合，振り替え受講者には機械をさわらせない。複数クラスが同時進行している場合には，講師間で意思統一して，１週間通しで同じ講義内容にする。月曜第１回の講師は必ずベテランをあてて，できれば他の講師も受講する態度が望ましい。
(10) 機械台数が少ない場合には，講師がこまめに交替を指示すること。
(11) 受講者のレベルによっては，第７回あたりから図形の時間が足りなくなる。そのときには，第１４回，第１５回の余裕をにらんで，適宜スケジュールの調整をする。
(12) ＯＡのための企業内講座の場合には，このＢＡＳＩＣ講座が逆効果になって『日暮れて道遠し』と思わせてはいけない。いわゆる汎用事務計算パッケージプログラムを用意しておいて，高級電卓程度の仕事は簡単にできることを認識させる。
(13) 『知っていること』と『教えること』とは違う。やさしく説明できる技術は事前の十分な練習によってしか身につかない。しゃべり過ぎに注意する。
(14) 受講者は，講師の想像以上に神経を使っている。早め早めに小休止をいれて背伸び，首の運動，指の運動などをする。
(15) 受講者の記録は必ずとる。どこで行き詰まったかをみると，講師の力不足が原因だと気づく。謙虚に反省して次回に備えよう。

テキスト　第1回

① ＰＣのテスト法

ＰＲＩＮＴ　ＦＲＥ（0）　…………16Kの場合10402とでる。
32Kの場合26786とでる。

ＢＥＥＰ1……………………ブザーが鳴る。
ＢＥＥＰ0……………………とまる。

ＰＲＩＮＴ　ＴＩＭＥ＄………スイッチを入れてからの時間がでる。

② ダイレクトモード

ＰＲＩＮＴ”ＡＳＡＨＩ”

ＰＲＩＮＴ　2＋3，6－3，5＊3，6／3

ＰＲＩＮＴの省略記号は疑問符（？）

ＰＲＩＮＴ文中のコンマ（，）とセミコロン（；）のちがい

③ プログラムモード

```
10　INPUT　A
20　PRINT　A
30　END
```

```
10　INPUT　A
20　LET　B=A
30　PRINT　B
40　END
```

```
10　INPUT　A,B
20　LET　C=A*B
30　PRINT　"A*B=";C
40　END
```

④ 図形　線を描く

```
LINE(20,0)-(20,15),"A",0
LINE(0,12)-(35,12),"◆",0
LINE(5,7)-(30,20),"*",0
```

テキスト　第2回

① ↑↓→　←　INSキー　DELキーの使いかた

② 円の面積

```
10 INPUT R
20 M=3.14159*R*R
30 PRINT M
40 END
```

```
10 INPUT R
20 M=3.14159*R^2
30 PRINT M
40 GOTO 10
50 END
```

```
10 INPUT R
20 IF R=0 THEN GOTO 60
30 M=3.14159*R*R
40 PRINT M
50 GOTO 10
60 END
```

③ STOP

```
10 INPUT R
20 A=R*R
30 STOP
40 M=3.14159*A
50 PRINT M
60 END
```

上のプログラムは30行でとまる。ダイレクトモードでPRINT A を実行するとR*Rの答が出る。CONTと入れてRETURNキーを押すとプログラムの実行が40行から再開される。STOPはデバック（Debug）などで使う。

④ 図形　箱を描く

```
10 LINE(5,5)-(35,20),"*",B
10 LINE(5,5)-(35,20),"*",BF
```

```
10 WIDTH 80,25
20 LINE(30,10)-(100,90),PSET,0
30 LINE(35,15)-(80,75),PSET,B
40 LINE(40,20)-(70,65),PSET,BF
```

テキスト　第3回

① 平均

```
10 A=1+2+3+4+5
20 PRINT A/5
30 END
```

```
10 INPUT A, B, C, D, E
20 PRINT (A+B+C+D+E)/5
30 GOTO 10
```

② ゼロを入れると平均を出力する

```
10 K=0:S=0
20 INPUT "データ ヲ イレナサイ";X
30 IF X=0 THEN 70
40 K=K+1
50 S=S+X
60 GOTO 20
70 PRINT"データ スウ=";K
80 PRINT"ゴウケイ=";S
90 PRINT"ヘイキン=";S/K
100 PRINT
110 END
```

③ 繰り返し

a.

```
10 X=1
20 PRINT X;
30 X=X+1
40 IF X<=30 THEN 20
50 END
```

b.

```
10 FOR X=1 TO 30
20 PRINT X;
30 NEXT X
40 END
```

aとbの結果は，同じになる。bの10行をFOR X =1 TO 30 STEP 2 としてみよ。

④ 図形　X軸とY軸を引き，任意の位置に点（ドット）を出す

```
10 WIDTH 80, 25
20 LINE (80, 0) - (80, 79), PSET, 0
30 LINE (0, 40) - (157, 40), PSET, 0
40 INPUT X
50 INPUT Y
60 PSET (X, Y, 0)
70 GOTO 40
```

テキスト　第4回

① １００から１まで３ずつ減らしていく

```
10 FOR J=100 TO 1 STEP -3
20 PRINT J;
30 NEXT J
40 END
```

② 九九の掛け算

```
10 FOR K=1 TO 9
20 FOR J=1 TO 9
30 PRINT J*K;
40 NEXT J
50 PRINT
60 NEXT K
70 END
```

③ READとDATA

```
10 READ X, Y, Z
20 A=X+Y+Z
30 PRINT X;Y;Z;A
40 DATA 7, 5, 3
50 END
```

READ文の変数の数とDATA文のデータの数と一致していることに注意。

④ 二次方程式

```
10 FOR X=-6 TO 6
20 PRINT TAB (X*X) ;"●"
30 NEXT X
40 END
```

⑤ 図形　点（ドット）が進む

```
10 WIDTH 80, 25
20 FOR X=-40 TO 40
30 Y=X
40 FOR J=1 TO 5
50 PSET (X+80, Y+40, 0)
60 NEXT J
70 PRESET (X+80, Y+40)
80 NEXT X
90 GOTO 20
```

テキスト　第5回

① 2つの数を入れて足し算をしてみる

```
 10 INPUT A, B
 20 GOSUB 100
 30 PRINT A+B
 40 IF (A+B)<100 THEN 80
 50 FOR J=1 TO 800
 60 BEEP1:BEEP0 '100>=(A+B)ノ トキ
 70 NEXT J
 80 GOSUB 100
 90 END
100 REM******SUBROUTINE******
110 BEEP1
120 FOR K=1 TO 800:NEXT K
130 BEEP0
140 RETURN
```

② RESTORE

```
10 READ A, B, C
20 READ D, E, F
30 PRINT A+B, D+E, C+F
40 DATA 2, 4, 6
50 DATA 2, 4, 6
60 END
```

```
10 READ A, B, C
20 RESTORE
30 READ D, E, F
40 PRINT A+B, D+E, C+F
50 DATA 2, 4, 6
60 END
```

RESTORE90などのように，行指定もできる。

③ 図形 ●で三角を描く

```
10 FOR J=1 TO 6
20 PRINT J ;TAB (4) ;"カイ";TAB (9) ;STRING$ (J,"●")
30 NEXT J
40 K=6
50 FOR J=6 TO 1 STEP-1
60 K=K+1
70 PRINT K ;TAB (4) ;"カイ";TAB (9) ;STRING$ (J,"●")
80 NEXT J
```

テキスト　第6回

① 変数

```
10 A$="ASAHI"
20 B$="WEEKLY"
30 PRINT A$+B$
```

```
10 INPUT R#
20 M#=3.14159*R#^2
30 PRINT M#
```

② 配列（添字つき変数）

```
10 DIM X(10)
20 FOR J=1 TO 10
30 X(J)=J*2
40 NEXT J
50 FOR K=1 TO 10
60 PRINT X(K);
70 NEXT K
```

DIM X(10)でX(0), X(1), ……X(10)まで11個の場所が確保される。上の例では, X(0)は使っていない。

```
10 INPUT N
20 DIM X(N)
30 FOR J=1 TO N
40 PRINT "X=";
50 INPUT X(J)
60 NEXT J
70 PRINT
80 T=0
90 FOR J=1 TO N
100 T=T+X(J)
110 NEXT J
120 PRINT "GOKEI=";T
```

③ 図形　Kの長さの線が進む

```
10 WIDTH 80,25
20 FOR X=-40 TO 40:Y=X
30 FOR K=0 TO 7:PSET(X+80+K,Y+40+K,0)
40 NEXT K
50 PRESET(X+80,Y+40,0):NEXT X
60 FOR K=0 TO 7
70 PRESET(120+K,80+K,0)
80 NEXT K
```

テキスト　第7回

① 2重添字つき変数

```
 10 DIM A(2, 3)
 20 FOR X=0 TO 2
 30 FOR Y=0 TO 3
 40 READ A(X, Y)
 50 NEXT Y
 60 NEXT X
 70 FOR X=0 TO 2
 80 FOR Y=0 TO 3
 90 PRINT A(X, Y)
100 NEXT Y
110 NEXT X
120 DATA1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22
```

60行までが実行されると，下のように読み込まれる。

A(0, 0) 1	A(0, 1) 2	A(0, 2) 4	A(0, 3) 6
A(1, 0) 8	A(1, 1) 10	A(1, 2) 12	A(1, 3) 14
A(2, 0) 16	A(2, 1) 18	A(2, 2) 20	A(2, 3) 22

70行と80行，100行と110行を入れ換えてみよ。

② 図形　1歩後退2歩前進

```
10 WIDTH 80, 25
20 FOR X=-40 TO 40
30 Y=X
40 FOR K=0 TO 4
50 PSET (X+80+K, Y+40+K, 0)
60 FOR J=1 TO 10:NEXT J
70 PRESET (X+80+K, Y+40+K, 0)
80 NEXT K
90 NEXT X
```

テキスト　第8回

① 作表

	サトウ	タナカ	キムラ	モリタ
エイゴ	56	94	47	82
コクゴ	72	63	93	79
メンセツ	83	58	71	67

```
 10 DIM N$(4),K$(2),A(2,3)
 20 DATA "      ","  サトウ","  タナカ","  キムラ"
 30 DATA "  モリタ"
 40 DATA "エイゴ","コクゴ","メンセツ"
 50 DATA 56,94,47,82,72,63,93,79,83
 60 DATA 58,71,67
 70 FOR J=0 TO 4:READ N$(J):NEXT J
 80 FOR J=0 TO 2:READ K$(J):NEXT J
 90 FOR X=0 TO 2:FOR Y=0 TO 3
100 READ A(X,Y):NEXT Y:NEXT X
110 FOR J=0 TO 4
120 PRINT USING "&   &";N$(J);:NEXT J
130 PRINT
140 FOR X=0 TO 2
150 PRINT USING "&   &";K$(X);
160 FOR Y=0 TO 3
170 PRINT USING "####";A(X,Y);:NEXT Y
180 PRINT
190 NEXT X
```

② RND（乱数）とINT（整数にする）で◆やサイコロの目を出す

```
10 N=INT(15*RND(1)+1)
20 PRINT STRING$(N,"◆")
30 GOTO 10
```

```
10 FOR J=1 TO 10
20 PRINT INT(6*RND(1)+1);
30 NEXT J
```

テキスト　第9回

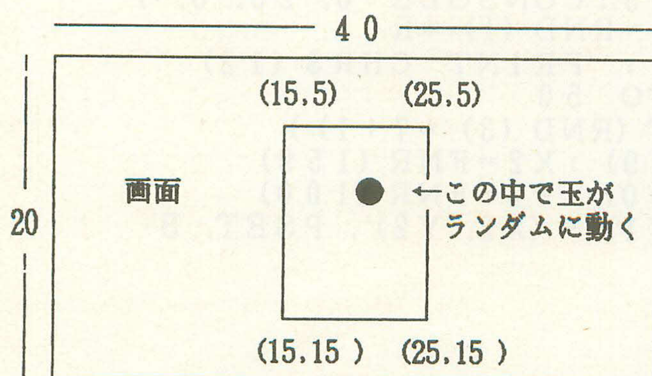
① 検索　コードで名前を探す

```
 10 DIM S(20), S$(20)
 20 GOSUB 200
 30 INPUT "コード ヲ ドウゾ";S
 40 FOR J=1 TO 20
 50 IF S(J)=S THEN 100
 60 NEXT J
 70 PRINT
 80 PRINT "コード ナシ"
 90 GOTO 30
100 PRINT "コード";S;" ナマエ ";S$(J)
110 PRINT
120 GOTO 30
130 END
200 FOR J=1 TO 10
210 READ S(J), S$(J)
220 NEXT J
230 RETURN
300 DATA 100, サトウ, 200, タナカ, 300, キムラ, 400
310 DATA コイケ, 500, アラキ, 600, ウスイ, 700, トミタ
320 DATA 800, ムライ, 900, カネコ, 910, ナカノ
```

② 名前でコードを探すには, 次のように変更する

```
 30 INPUT "ナマエ ヲ ドウゾ";S$
 50 IF INSTR(S$(J), S$)<>0 THEN 100
100 PRINT"ナマエ ";S$(J);" コード";S(J)
```

③ 図形　箱の中で玉が動く　プログラムは次のページにある



③のつづき

```
 10 REM *****ハコ ノ ナカ デ タマ ガ ウゴク *****
 20 LINE(15, 6)-(15,14),"|"
 30 LINE(25, 6)-(25,14),"|"
 40 LINE(16, 5)-(24, 5),"—"
 50 LINE(16,15)-(24,15),"—"
 60 LOCATE15,5:PRINT"┌"
 70 LOCATE15,15:PRINT"└"
 80 LOCATE25,5:PRINT"┐"
 90 LOCATE25,15:PRINT"┘"
100 X=20
110 Y=10
120 LOCATE X,Y:PRINT"●"
130 BEEP1:FOR J=1 TO 50:BEEP0:NEXT J
140 LOCATE X,Y:PRINT" "
150 FOR J=1 TO 20:NEXT J
160 X=X+INT(RND(3)*3)-1
170 Y=Y+INT(RND(3)*3)-1
180 REM *****ハコ カラ デナイヨウニスル*****
190 IF X<16 THEN X=X+1
200 IF X>24 THEN X=X-1
210 IF Y<6  THEN Y=Y+1
220 IF Y>14 THEN Y=Y-1
230 GOTO 120
```

次の行を追加すると，XとYの位置（玉の位置）がわかる。

```
185 LOCATE1,1:PRINT X,Y
```

④ 図形 ランダムに箱を描く （関数の定義は第12回でやる）

```
 10 WIDTH80,25:CONSOLE 0,25,0,1
 20 DEFFNR(R)=RND(1)*R
 30 COLOR,0,1: PRINT CHR$(12)
 40 FOR J=1 TO 50
 50 COLOR(INT(RND(3)*7+1))
 60 X1=FNR(159):X2=FNR(159)
 70 Y1=FNR(100):Y2=FNR(100)
 80 LINE(X1,Y1)-(X2,Y2),PSET,B
 90 NEXT J
100 END
```

テキスト　第10回

① 分類（ならべかえ）

```
 10 DIM S(10), S$(10)
 20 GOSUB 200
 30 FOR J=1 TO 9
 40 FOR K=1 TO 10-J
 50 IF S(K)<=S(K+1) THEN 80
 60 SWAP S(K), S(K+1)
 70 SWAP S$(K), S$(K+1)
 80 NEXT K
 90 GOSUB 300
100 NEXT J
110 END
200 FOR L=1 TO 10
210 READ S(L), S$(L)
220 NEXT L
230 RETURN
240 DATA 3, C, 6, F, 10, J, 4, D, 2, B, 7, G
250 DATA 5, E, 1, A, 9, I, 8, H
300 FOR M=1 TO 10
310 PRINT S(M);S$(M);
320 NEXT M
330 PRINT
340 RETURN
```

SWAPを使わなければ，次のようになる。

```
 60 T=S(K):T$=S$(K)
 65 S(K)=S(K+1):S$(K)=S$(K+1)
 70 S(K+1)=T:S$(K+1)=T$
```

② 最小値を求める

```
 10 INPUT "カズ ノ コスウ";N
 20 DIM X(N)
 30 FOR J=1 TO N
 40 INPUT "X=";X(J)
 50 NEXT J
 60 PRINT
 70 M=X(1)
 80 FOR K=1 TO N
 90 IF M <= X(K) THEN 110
100 M=X(K)
110 NEXT K
120 PRINT "サイショウチ=";M
130 END
```

テキスト　第11回

① 関数

```
10 FOR J=1 TO 40
20 PRINT SQR(J), RND(J), EXP(J)
30 NEXT J
40 END
```

② 円を描く

```
10 WIDTH 80, 25
20 P=3.14159
30 FOR T=0 TO 2*P STEP .03
40 X=30*COS(T)+40
50 Y=30*SIN(T)+50
60 LINE(X, Y)-(40, 50), PSET
70 X1=X+80:Y1=Y
80 PSET(X1, Y1, 0)
90 NEXT T
100 END
```

40行を　X=40*COS(T)+40　にしてみよ。

③ うずまきを描く

```
10 WIDTH 80, 25
20 FOR T=1 TO 1440
30 X=COS(T*3.14159/180)
40 Y=SIN(T*3.14159/180)
50 PSET(T/40*X+80, T/40*Y+40, 0)
60 NEXT T
70 END
```

④ テンキーの2, 4, 6, 8を使って線を描く　（キーボードマップ参照のこと）

```
10 WIDTH 80, 25:COLOR, 0, 1
20 PRINT CHR$(12):X=80:Y=50
30 PSET(X, Y)
40 IF INP(0)=251 THEN Y=Y+1
50 IF INP(0)=239 THEN X=X-1
60 IF INP(0)=191 THEN X=X+1
70 IF INP(1)=254 THEN Y=Y-1
80 IF INP(1)=127 THEN END
90 GOTO 30
```

テキスト　第12回

①　関数の自由定義

円の面積は第２回でやったとおり　M＝3．14159＊R＊R　だった。
MをRの関数という。ここで，関数関係を明確にするために次のようにする。

```
DEFFN M(R)=3.14159*R*R
```

使うときには，FN　M（式）の形で使う。

```
10 DEFFN M(R)=3.14159*R*R
20 X=FN M(5)
30 A=10
40 Y=FN M(A)
50 Z=FN M(A+5)
60 PRINT X,Y,Z
70 END
```

②　元利合計を求める

```
10 DEFDBL N-S
20 DEFFNS(P,N,R)=P*(1+R)^N
30 INPUT "ガンキン=";P
40 INPUT "リリツ=";R
50 INPUT "ネンスウ=";N
60 PRINT
70 PRINT"ガンリゴウケイ=";FNS(P,N,R)
80 END
```

③　乱数を発生させて条件で行き先をかえる

```
10 WIDTH 80,25: PRINT CHR$(12)
20 IF A =75 OR B =75 OR C =75 OR D=75 THEN 500
30 R=INT(4*RND(1)+1)
40 ON R GOTO 100,200,300,400
100 A =A+1 :LOCATE 0,0 :PRINT STRING$(A,"●"):GOTO20
200 B =B+1 :LOCATE 0,4 :PRINT STRING$(B,"○"):GOTO20
300 C =C+1 :LOCATE 0,8 :PRINT STRING$(C,"◆"):GOTO20
400 D =D+1 :LOCATE 0,12:PRINT STRING$(D,"■"):GOTO20
500 LOCATE 0,16:PRINT MID$("●○◆■",R,1);"ノカチ"
```

④　乱数の作りかたの例

1と－1をつくる	INT(RND(3)＊2)＊2－1
1と0と－1を出す	INT(RND(1)＊3)－1
25から54を出す	INT(RND(1)＊30)＋25

テキスト　第13回

① 2進・10進・16進

10進数　2　進数　16進数

```
 0……0000……0
 1……0001……1
 2……0010……2
 3……0011……3
 4……0100……4
 5……0101……5
 6……0110……6
 7……0111……7
 8……1000……8
 9……1001……9
10……1010……A
11……1011……B
12……1100……C
13……1101……D
14……1110……E
15……1111……F
```

8の位 ── 4の位 ── 2の位 ── 1の位

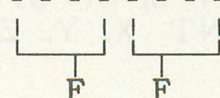
(1) 2進数の1桁をビットという。

(2) 2進数のビットをぜんぶ足すと10進数になる。

0111＝0＋4＋2＋1＝7

(3) 2進数8桁の最大は

11111111＝255＝FF



② 10進数を2進数に変換する

```
 10 DIM A(16):B=0:C=0:J=0
 20 INPUT "10シンスウ=";D
 30 T=D
 40 B=T¥2:C=T MOD 2
 50 A(J)=C
 60 IF B=1 THEN A(J+1)=1:GOTO 90
 70 T=B
 80 IF B =0 THEN GOTO 90 ELSE J =J+1 :GOTO 40
 90 PRINT D;"=";
100 FOR K=J+1 TO 0 STEP -1
110 PRINTUSING"#";A(K);
120 NEXT K
130 PRINT:ERASE A:GOTO 10
```

③ 1文字をコードにコードを文字に変換する（キャラクターコード表参照）

```
10 PRINT ASC("B")
20 PRINT CHR$(65)
30 FOR J =45 TO 255 :PRINT CHR $ (J ) ; :NEXT J
40 END
```

テキスト　第14回

① 折れ線グラフ

```
10  WIDTH 80,25 : COLOR 0,0,1
20  DIM AX (145 ) , AY (90)
30  INPUT "データ ノ コスウ";KX                  データの個数
40  INPUT "データ ノ サイショウチ";YS              縦軸の最小値
50  INPUT "データ ノ サイダイチ";YE               縦軸の最大値
60  INPUT "メモリ ノ ハバ";KY                    縦軸の目盛りの幅
70  JX= (146 /KX)                               横軸の画面上のきざみ
80  IF JX <2 THEN 30
90  IY= (YE―YS) /KY                            縦軸の目盛りの個数
100 JY=90/ (YE―YS)                             縦軸の画面上のきざみ
110 IF 90/KY <2 THEN 40
120 PRINT CHR$ (12)                             画面クリアー
130 LINE (10,90 ) ― (159,90) ,PSET              横軸を描く
140 LINE (10,90 ) ― (10,0) ,PSET                縦軸を描く
150 PX=10
160 FOR J=1 TO KX                               横軸に10からはじめてJX
170 PX=PX+JX                                    の幅できざみ点をつける
180 PSET (PX,91 )
190 NEXT J
200 FOR J=1 TO IY                               縦軸に10からはじめて
210 PY=PY + JY * KY                             JY＊KYの幅で点と目盛
220 PSET (9,90―PY)                              りをつける
230 LOCATE 0, (90―PY) / 4
240 PRINT USING "###";YS + KY * J
250 NEXT J
260 PX=10 + JX
270 FOR J=1 TO KX
280 LOCATE 60,1
290 INPUT "データ";Y                             データの入力
300 YY=90― JY * (Y ―YS)
310 PSET (PX,YY )                               点をうつ
320 AX (J ) =PX                                 点の位置を保存する
330 AY (J ) =YY
340 PX=PX + JX                                  次のきざみへいく
350 NEXT J
360 FOR J=2 TO KX                               点を線でつなぐ
370 LINE (AX (J ―1 ) , AY (J ―1 ) ) ― (AX (J ) ,AY (J ) ) ,PSET
380 NEXT J
390 END
```

３９０　GOTO 260　とすると複数の線を描くことができる。

テキスト　第15回

① ヒストグラム

```
10  WIDTH 80,25 :PRINT CHR$ (12)
20  DIM D (10)
30  LOCATE 5,0:PRINT "**HISTGRAM**"
40  FOR I =2 TO 13
50    LOCATE 15,I :PRINT "|"                     縦の線
60    LOCATE 40,I :PRINT "・"
70    LOCATE 65,I :PRINT "・"
80  NEXT I
90  FOR I =15 TO 70
100   LOCATE I,13 :PRINT "—"                     横の線
110 NEXT I
120 L =3
130 FOR I =1 TO 10
140   LOCATE 0,L :PRINT "rank";I                 ランク
150   LOCATE 8,L :PRINT USING "##"; D (I )
160   L =L+1
170 NEXT I
180 LOCATE 0,15 : PRINT "total "                 合計
190 LOCATE 7,15 :PRINT USING "##"; D (0 )
200 FOR I =0 TO 50 STEP 10                       横の刻み,0,10,20..50
210   LOCATE I+14 ,14 :PRINT USING "##";I
220 NEXT I
230 LOCATE 0,20 :PRINT "←スペース11個→"          データのインプットは
240 LOCATE 0,20 :INPUT "DATA"; X$                ENDで終わり
250 IF X$="end" THEN 370
260 X = VAL ( X$)                                X>100,X <0 は
270 IF X>=0 AND X <=100 THEN 290                 データエラー
280 LOCATE 0,21 :PRINT "DATA error" :GOTO 230
290 LOCATE 0,21 :PRINT "←スペース17個→"
300 IF X=100 THEN J=10 :GOTO 320
310 J = INT (X /10) + 1
320 D (J ) = D (J ) +1: D (0 ) = D (0 ) +1       インプットした値を
330 LOCATE 8,J+2:PRINT USING "##"; D (J )        配列の各ランクに
340 LOCATE 15+ D (J ) , J+2 :PRINT "■"
350 LOCATE 7,15 :PRINT USING "###"; D (0)
360 GOTO 230
370 END
```

```
234  FOR K=1 TO 200
235  X = (RND (1 ) +RND (2 ) +RND (3 ) ) /3 *100 :GOTO 270
```

を追加し

```
360  NEXT K
```

と変更すると自動的に正規分布ヒストグラムを作る。

ディスクを使ったプログラム例

＜例１＞

＊機能

指定した名前のファイルを複写し，新たに指定した名前のファイルを作る。ランダムファイルでもシーケンシャルファイルでもよい。

＊ポイント

ＦＩＥＬＤ文に注意。文字変数の最大長は２５５バイトなので，ＦＩＥＬＤ文に１つの文字変数しか定義しない場合，１バイト削られてしまう。

```
10 '**********************
20 '*                    *
30 '*     ファイル コピー    *
40 '*                    *
50 '**********************
60 PRINT CHR$(12);
70 PRINT "*** ファイル メイ ヲ イレテ クダサイ"
80 INPUT "インプット";S$
90 INPUT "アウトプット";D$
100 OPEN S$ AS #1
110 OPEN D$ AS #2
120 FIELD #1,128 AS S1$,128 AS S2$
130 FIELD #2,128 AS D1$,128 AS D2$
140 IF LOC(1)=LOF(1) THEN 200
150   GET #1
160   LSET D1$=S1$
170   LSET D2$=S2$
180   PUT #2
190 GOTO 140
200 PRINT "end"
210 CLOSE
220 END
```

＜例２＞

＊機能

シーケンシャルファイルの名前と調べたい文字列を入力すると，同じ文字列があるレコードの内容を画面に出す。終了するときは，ＲＥＴＵＲＮキーを押す。ファイルの１レコードは，２５６バイト以下であること。

＊ポイント

ＬＩＮＥＩＮＰＵＴ文で１レコードを文字変数として読む。

```
10 '**********************
20 '*                    *
30 '*    ストリング サーチ    *
40 '*                    *
50 '**********************
60 INPUT "ファイルメイ";F$
70 WD$="":INPUT "シラベル ストリング";WD$
80 OPEN F$ FOR INPUT AS #1
90 IF WD$="" THEN 160
100 'loop
110   IF EOF(1) THEN 150
120   LINEINPUT#1,L$
130   IF INSTR(L$,WD$) THEN PRINT L$
140   GOTO 100
150 CLOSE#1:GOTO 70
160 CLOSE#1
```

<例3>

＊機能

アスキー形式でSAVEされたプログラムA，Bを比較し，両者の違いを画面に出力する。比較のキーは，文番号。

＊ポイント

キーの大小比較によるシーケンシャルファイルの処理は，この形が基本。

```
10 '**********************
20 '*   プログラム ヒカク   *
30 '**********************
40 HV=99999
50 INPUT "file A:";A$
60 INPUT "file B:";B$
70 OPEN A$ FOR INPUT AS #1
80 OPEN B$ FOR INPUT AS #2
90 GOSUB 170
100 GOSUB 190
110 IF AK<BK THEN PRINT"A ノミ";AF$
                   :GOSUB 170:GOTO110
120 IF AK>BK THEN PRINT"B ノミ";BF$
                   :GOSUB 190:GOTO110
130 IF AK=HV THEN PRINT "オワリ"
                   :CLOSE:END
140 IF AF$<>BF$ THEN PRINT "A ワ ";AF$
                     :PRINT "B ワ ";BF$
150 GOSUB 170:GOSUB 190
160 GOTO 110
170 IF EOF(1) THEN AK=HV
     ELSE LINEINPUT#1,AF$:AK=VAL(AF$)
180 RETURN
190 IF EOF(2) THEN BK=HV
     ELSE LINEINPUT#2,BF$:BK=VAL(BF$)
200 RETURN
```

＜例４＞

＊機能

カーソルを動かして作った画面（キャラクターのみ）をディスクに保存する。プリンターに出力することもできる。

＊ポイント

画面の内容をＧＥＴ＠で配列に取り込み，ディスクに書き出す。

```
 10 '＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
 20 '＊  ガメン ファイル  (1)    ＊
 30 '＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
 40 WIDTH80,25:CONSOLE24,24,0:COLOR,32
 50 DEFUSR1=&H124A:DIM A%(99),A$(99)
 60 PRINT CHR$(12)
 70 LOCATE20,0:PRINT"＊＊＊ ガメン ツクリ ＊＊＊"
 80 FOR I=1 TO 2000:NEXT
 90 PRINT CHR$(12)
100 LOCATE 0,24
110 INPUT "オワッタラ RETURN キー";W$
120 LOCATE 0,24:PRINT
130 INPUT "プリント (y/n) ";W$:PRINT:PRINT
140 IF W$="y" THEN P=USR1(1)
150 INPUT "シュウセイ (y/n) ";W$:PRINT:PRINT
160 IF W$="y" THEN 100
170 INPUT "ファイル  (y/n) ";W$:PRINT:PRINT
180 IF W$="n" THEN 210
190 INPUT "ガメン ノ ナマエ";GN$:PRINT:PRINT
200 GOSUB 220
210 END
220 OPEN GN$ AS#1
230 FOR J=0 TO 99
240   FIELD#1,J*2 AS DM$,2 AS A$(J)
250 NEXT J
260 FOR I=0 TO 79 STEP 4
270   GET@A(I,0)-(I+3,24),A%
280   FOR J=0 TO 99
290     LSET A$(J)=MKI$(A%(J))
300   NEXT J
310   PUT#1
320 NEXT I
330 CLOSE#1
340 RETURN
```

＜例５＞

＊機能

例４で作ったファイルを読んで画面に出す。必要ならば修正を加えてディスクに保存する。プリンターに出力することもできる。

＊ポイント

ディスク上のファイルを読み，ＰＵＴ＠で画面にだす。

```
10 '*********************
20 '*  ｶﾞﾒﾝ ﾌｧｲﾙ    (2) *
30 '*********************
40 WIDTH80,25:CONSOLE24,24,0:COLOR,32
50 DEFUSR1=&H124A:DIM A%(99),A$(99)
60 PRINT CHR$(12)
70 INPUT"ｶﾞﾒﾝ ﾒｲ";GN$:PRINT CHR$(12)
80 GOSUB 210
90 LOCATE0,24:PRINT
100 INPUT"ﾌﾟﾘﾝﾄ (y/n)";W$:PRINT:PRINT
110 IF W$="y" THEN PRINT:P=USR1(1)
120 LOCATE0,24
130 INPUT "ｼｭｳｾｲ ｼﾏｽｶ (y/n)";W$:PRINT
140 IF W$="n" THEN 170
150 INPUT "ｵﾜｯﾀﾗ RETURN ｷｰ ";W$:PRINT
160 GOTO 90
170 INPUT"ﾌｧｲﾙ ｼﾏｽｶ (y/n)";W$:PRINT
180 IF W$="n" THEN 200
190 GOSUB 340
200 END
210 OPEN GN$ AS#1
220 FOR J=0 TO 99
230   FIELD#1,J*2 AS DM$,2 AS A$(J)
240 NEXT J
250 FOR I=0 TO 79 STEP 4
260   GET#1
270   FOR J=0 TO 99
280     A%(J)=CVI(A$(J))
290   NEXT J
300   PUT@A(I,0)-(I+3,24),A%
310 NEXT I
320 CLOSE#1
330 RETURN
340 INPUT"ｶﾞﾒﾝ ﾒｲ";GN$:PRINT:PRINT
350 OPEN GN$ AS#1
360 FOR J=0 TO 99
370   FIELD #1,J*2 AS DM$,2 AS A$(J)
380 NEXT J
390 FOR I=0 TO 79 STEP 4
400   GET@A(I,0)-(I+3,24),A%
410   FOR J=0 TO 99
420     LSET A$(J)=MKI$(A%(J))
430   NEXT J
440   PUT#1
450 NEXT I
460 CLOSE#1
470 RETURN
```

＜例６＞

＊機能

本の索引ファイルを作る。必要ならば，作ったファイルを分類してプリンターに出す。入力は，項目名，ページのどちらを先に入れてもよい。ページは頭にＰをつけて入れること。ドライブ２にデータファイルを作る。

＊ポイント

行番号３６０は，プログラムのチェイン。

```
10  CLEAR 10000
20  DIM CF$(101),AA$(401)
30  OPEN"2:INDEX.DAT" AS#1
40  FIELD#1,20 AS AF$,1 AS BF$
50  FOR I=1 TO 101
60    FIELD#1,54+(I-1)*2 AS DM$,
              2 AS CF$(I)
70  NEXT I
80  FS=LOF(1)
90  IF FS>0 THEN FOR I=1 TO FS:GET#1,I:
                 AA$(I)=AF$:NEXT I
100 'LOOP
110   WD$="":INPUT WD$
120   IF WD$="" THEN 340
130   IF NOT((LEFT$(WD$,1)="P" OR
         LEFT$(WD$,1)="p") AND
         VAL(MID$(WD$,2))>0) THEN 150
140     PG=VAL(MID$(WD$,2)):GOTO 330
150   'ELSE
160     WD$=LEFT$(WD$+SPACE$(20),20)
170     AA$(FS+1)=WD$
180     FOR I=1 TO FS+1
190       IF AA$(I)=WD$ THEN 210
200     NEXT I
210     IF NOT(I>FS)THEN 260
220       FS=FS+1
230       LSET AF$=WD$:LSET BF$=CHR$(1)
240       LSET CF$(1)=MKI$(PG)
250       PUT#1,FS:GOTO 330
260     'ELSE
270       GET#1,I:B=ASC(BF$)
280       LSET CF$(B+1)=MKI$(PG)
290       FOR J=1 TO B+1
300         IF CVI(CF$(J))=PG THEN 320
310       NEXT J
320     IF J>B THEN LSET BF$=CHR$(J):
                   PUT#1,I
330   GOTO 100
340 CLOSE#1
350 INPUT "プリント シマスカ(y/n)";YN$
360 IF YN$="y" THEN RUN "pinx"
```

<例7>

＊機能

例6で作ったファイルを読み，分類してプリンターへ出力する。データを追加することもできる。ドライブ2のデータファイルを読む。

＊ポイント

行番号60～140，150～260でそれぞれ項目，ページの分類をする。

```
 10 CLEAR 10000:DIM CF$(101),C%(101),
                        AA$(401),RN%(401)
 20 OPEN "2:INDEX.DAT" AS#1
 30 FIELD#1,20 AS AF$,1 AS BF$
 40 FOR I=1 TO 101:FIELD#1,54+(I-1)*2
    AS DM$,2 AS CF$(I):NEXT I
 50 FS=LOF(1):IF NOT(FS>0) THEN 300
 60   FOR I=1 TO FS:GET#1,I:L=1:R=I-1
 70     'LOOP
 80       IF L>R THEN 120
 90       M=(L+R)¥2
100       IF AA$(M)>AF$ THEN R=M-1
                          ELSE L=M+1
110       GOTO 70
120     IF L<=I-1 THEN
          FOR J=I-1 TO L STEP -1:
          AA$(J+1)=AA$(J):
          RN%(J+1)=RN%(J):NEXT J
130     AA$(L)=AF$:RN%(L)=I
140   NEXT I
150   FOR I=1 TO FS
160     GET#1,RN%(I):B=ASC(BF$)
170     FOR J=1 TO B
180       C=CVI(CF$(J)):L=1:R=J-1
190       'LOOP
200         IF L>R THEN 240
210         M=(L+R)¥2
220         IF C%(M)>C THEN R=M-1
                          ELSE L=M+1
230         GOTO 190
240       IF L<=J-1 THEN
            FOR K=J-1 TO L STEP -1:
            C%(K+1)=C%(K):NEXT K
250       C%(L)=C
260     NEXT J
270     LPRINT USING
        "&                    &";AF$;
280     FOR J=1 TO B:LPRINT USING
        "####";C%(J);:NEXT J:LPRINT
290   NEXT I
300 CLOSE#1
310 INPUT "ﾏﾀﾞ ﾄｳﾛｸｼﾏｽｶ(y/n)";YN$
320 IF YN$="y" THEN RUN "inx"
```

<例８>

＊機能

社員コードの順番に売り上げを入力し，ハンバイファイルを作る。次のプログラムでは，ハンバイファイルを読み，報告書を作る。報告書は，社員コードで売り上げ金額の小計，社員コードの頭１桁で中計，すべての社員の大計をとる。

＊ポイント

集計をとるプログラムは，この形が基本。報告書作成プログラムの行番号９０，１００で行っているキーが変わったときの処理に注意。

入力データ

社員コード	売り上げ金額
１１１	５，０００
１１１	４，０００
１１５	２，５００
２００	１，５００
２５０	２，０００
２５０	４，５００
３６０	５，５００
４１０	３，０００
４１０	７，０００
４５０	３，５００

出力結果

```
１１１      ５，０００
１１１      ４，０００
                        １１１ケイ      ９，０００
１１５      ２，５００
                        １１５ケイ      ２，５００
--------------------------------------------------
                      １ショウケイ     １１，５００
--------------------------------------------------
２００      １，５００
                        ２００ケイ      １，５００
２５０      ２，０００
２５０      ４，５００
                        ２５０ケイ      ６，５００
--------------------------------------------------
                      ２ショウケイ      ８，０００
--------------------------------------------------
３６０      ５，５００
                        ３６０ケイ      ５，５００
--------------------------------------------------
                      ３ショウケイ      ５，５００
--------------------------------------------------
４１０      ３，０００
４１０      ７，０００
                        ４１０ケイ     １０，０００
４５０      ３，５００
                        ４５０ケイ      ３，５００
--------------------------------------------------
                      ４ショウケイ     １３，５００
--------------------------------------------------
                   ソウゴ゛ウケイ      ３８，５００
--------------------------------------------------
```

```
 10 '****************
 20 '* データ ニュウリョク *
 30 '****************
 40 OPEN "ハンバイ" FOR OUTPUT AS #1
 50 INPUT"シャインコード";CD$
 60 IF CD$="999" THEN GOTO 100
 70 INPUT"ウリアゲ";UR
 80 PRINT #1,CD$;",";UR
 90 GOTO 50
100 CLOSE#1
110 END
```

```
 10 '****************************
 20 '* ホウコクショ サクセイ (シュウケイ ノ シカタ) *
 30 '****************************
 40 OPEN "ハンバイ" FOR INPUT AS #1
 50 T1=0:T2=0:T3=0:SW=0
 60 IF EOF(1) THEN GOTO 140
 70 INPUT#1,CD$,UR
 80 IF SW=0 THEN SW=1:SV$=CD$
 90 IF LEFT$(CD$,1)<>LEFT$(SV$,1) THEN
    GOSUB 170:GOSUB 220:SV$=CD$
100 IF CD$<>SV$ THEN GOSUB 170:SV$=CD$
110 LPRINT USING "& &  ###,###";CD$;UR
120 T1=T1+UR
130 GOTO 60
140 GOSUB 170:GOSUB 220:GOSUB 300
150 CLOSE#1
160 END
170 'ショウケイ ノ プリント
180 LPRINT TAB(12);SV$;"ケイ";
190 LPRINT USING "###,###";T1
200 T2=T2+T1:T1=0
210 RETURN
220 'チュウケイ ノ プリント
230 LPRINT STRING$(30,"-")
240 LPRINT TAB(11);LEFT$(SV$,1);
250 LPRINT "ショウケイ";
260 LPRINT USING"###,###";T2
270 LPRINT STRING$(30,"-")
280 T3=T3+T2:T2=0
290 RETURN
300 'ダイケイ ノ プリント
310 LPRINT TAB(10);"ソウゴウケイ";
320 LPRINT USING"###,###";T3
330 LPRINT STRING$(30,"-")
340 RETURN
```

ＰＣ－６００１の特長

(1) ＰＣ－６００１のＢＡＳＩＣもＰＣ－８００１と同様にマイクロソフト社のＢＡＳＩＣを土台にしている。しかし，ＰＣ－８００１と内部コードは異なる。

(2) キャラクターコードも「ひらがな」を追加したことにより，ＰＣ－８００１とは異なる。

(3) 音楽機能が追加された。ＳＯＵＮＤ，ＰＬＡＹを見よ。グラフィック機能が強化された。ＰＡＩＮＴ，ＣＯＬＯＲ，ＬＩＮＥなどを見よ。

(4) 複数のページを持っているので，テキスト画面とグラフィック画面などを別々に持つことができる。ＳＣＲＥＥＮを見よ。

(5) ＲＯＭカートリッジ，ＲＯＭ＆ＲＡＭカートリッジを装着できる。ゲーム用にジョイスティックが２個つけられる。ディジタイザーも使える。オーディオ出力が可能になった。

(6) 家庭用テレビに直接つなぐことができる。アダプターは必要ない。

(7) オプションでＲＳ－２３２Ｃボードが用意されている。

(8) キーボードにオーバーレイシートを使用した。標準では，ひらがなのオーバーレイシートがついている。グラフィックやカタカナのシートと取り替え可能。

ＰＣ－６００１の扱い方

(1) ＰＣ－６００１に接続されている機器の電源を入れた後にＰＣ－６００１の電源を入れる。

(2) ＰＣ－６００１の画面に　Ｈｏｗ　Ｍａｎｙ　Ｐａｇｅｓ？と表示されるので，使用する画面のページ数を入れる。ＲＯＭ＆ＲＡＭカートリッジがある場合４ページ，ない場合２ページが最大ページ数。ページ数の指定をせずにＲＥＴＵＲＮキー押すと最大ページ数がとられる。

(3) テレビにつないだときは，チャンネルを１または２（その地域で使用していないチャンネル）にする。ＰＣ－６００１のチャンネルは，ＲＦ出力端子の下にある青いスイッチで切り換える。

(4) ＲＯＭ＆ＲＡＭカートリッジを使用するときは，カートリッジ挿入孔に入れる。ＰＣ－６００１の電源が入っている状態で挿入すると，ＰＣ－６００１の電源が１度切れた後で再び入る。ＲＯＭのプログラムを実行するときは，この後でＲＥＴＵＲＮキーを押す。ＲＯＭ＆ＲＡＭカートリッジを使用しているときにプログラムをカセットテープに保存した場合には，プログラムを読み込むときにも必ずＲＯＭ＆ＲＡＭカートリッジをつけておくこと。

PC－6001 キー 配 置 図

ファンクションキー SHIFT+F1→SCREEN　SHIFT+F2→CSAVE "　SHIFT+F3→PRINT
の定義 ――――――→F1のみ――→COLOR　F2のみ――→CLOAD "　F3のみ――→GOTO

ファンクションキー→		F 1		F 2		F 3		F 4			
ESC	! 1	" 2	# 3	$ 4	% 5	& 6	' 7	(8			
TAB	Q	W	E	R	T	Y	U				
CTRL	A	S	D	F	G	H	J				
SHIFT	Z	X	C	V	B	N					
		GRAPH	スペースキー								

（1）キー

①
②
③

キーを押すと，③の位置の文字が出る。英字の場合は②の位置の小文字がでる。SHIFT を押してキーを押すと①の位置の文字が出る。英字の場合は②の位置の大文字が出る。

（2）TABキー
8カラムごとにカーソルを右へ移動する。

（3）ESCキー
プログラムの一時停止。再開はキーボード上のどのキーを押してもよい。

（4）CTRLキー
スクリーンエディットのときに使う。このキーを押しながら次のキーを押す。

B→カーソルを1項目ごとに左にずらす。
C→STOPキーと同じ。
E→カーソルから行の終わりまでを消す。
F→カーソルを1項目ごとに右にずらす。
G→BEEPと同じ。
H→1文字削除。
I→TABキーと同じ。
J→ラインフィード。
K→HOMEキーと同じ。
L→CLRキーと同じ。
M→RETURNキーと同じ。
R→INSキーと同じ。
U→カーソルのある行を1行抹消する。

(5) STOPキー
プログラムの強制停止。

(6) CLR →押すと画面クリアー。
HOME→SHIFT+CLR カーソルが左上へ。
INS →押すとカーソルの左に文字等を挿入できる。解除はINS,RETURN,DEL, カーソル移動のいずれかのキーを押す。
DEL →押すとカーソルの左1文字を削除。

SHIFT+F4→PLAY　SHIFT+F5→CONT+RETURN
F4のみ──→LIST　F5のみ──→RUN+RETURN

F4　F5　HOME/CLR　STOP　INS　DEL
) 9　0　= -　^　¥　↰ ↳ ←画面切り替えキー
I　O　P　@　[　RETURN　↑
K　L　+ ;　* :　]　か な　← →
M　< ,　> .　? /　_　SHIFT　↓

カーソル移動。

(7) RETURNキー
入力の終わりに必ず押すこと。

(8) スペースキー
このキーを押すと空白がひとつとられる。

(9) かなキー
押すとロックされてオーバーレイシートに書かれているひらがなになる。解除はもう一度押す。SHIFT キーを押しながら画面切り替えキーを押すとカタカナに。

(10) 画面切り替えキー
画面表示するページを切り替える。画面の左下に表示画面のページ数が出る。

STOP
日　月　火　水　木　金　土　百　千　万　一　円
┌　┬　┐　年　│　π　○　RETURN
秒　├　┼　┤　時　中　♣　♥　●
╳　└　┴　┘　分　小　大　♠　◆

GRAPH →
キー

上の文字および図形を出すには, GRAPH キーを押しながら各キーを押す。

PC－6001 文字・演算子

機能	&H 疑問符（?） コロン（:） セミコロン（;） コンマ（,） 引用符（”）
型宣言	$
算術演算	＝ － ＋ ＊ ／ ^
論理演算	NOT AND OR
関係演算	＝ < > <= =< >= => <> ><

PC－6001 コマンド・ステートメント一覧表

コマンド	LIST LLIST NEW CONT RUN CLOAD CLOAD? CSAVE EXEC CLEAR
ステートメント	REM END FOR～NEXT GOSUB GOTO ON～GOTO ON～GOSUB RETURN STOP IF～THEN LET DIM DATA READ RESTORE POKE SOUND PLAY DEFFN IF～GOTO
入出力ステートメント	INPUT PRINT LPRINT INPUT# PRINT# LCOPY OUT
画面ステートメント	CONSOLE COLOR CLS LOCATE PSET PRESET LINE PAINT SCREEN
特殊ステートメント	KEY
数値関数	ABS COS EXP INT LOG RND SGN SIN SQR TAN
ストリング関数	CHR$ LEFT$ MID$ RIGHT$ STR$ ASC LEN VAL INKEY$
入出力関数	INP STICK STRIG
その他の関数	CSRLIN POS LPOS PEEK FRE SPC TAB TIME POINT USR

ＰＣ－８００１と機能がまったく同じものについては，説明を省略した。１部機能が異なるものについては，その部分だけ説明した。
プログラムの１行に認められる命令の長さおよびＩＮＰＵＴ文で入力できるデータの長さは，７１文字まで。
数値は有効桁数９桁。数値の範囲は，３×１０＾－３８から１．７×１０＾３８まで。
文字列に入る文字の数は，２５５文字まで。

（1） コマンド

ＬＩＳＴ〔<開始行番号>〕〔－〔<終了行番号>〕〕

１５ページ参照。ＬＩＳＴ．は使えない。

ＣＬＯＡＤ”<ファイル名>”

１４ページ参照。ファイル名を指定しないと最初に見つけたプログラムを読み込む。

ＥＸＥＣ<番地>

指定した番地からの機械語プログラムを実行する。

ＥＸＥＣ＆ＨＤ１００……………Ｄ１００番地からの機械語プログラムを実行する。

機械語プログラム実行後ＢＡＳＩＣに戻るためには，機械語プログラムの最後をＲＥＴ（１６進のＣ９）とする。機械語プログラムの書き込みは，ＰＯＫＥで行う。

（2） ステートメント

ＩＦ<条件式>ＴＨＥＮ<文または行番号>

２１ページ参照。ＥＬＳＥは使えない。

ＤＥＦＦＮ（<引数>）

１９ページ参照。引数は１種類しか使えない。

ＳＯＵＮＤ　<レジスター番号>，<データ>

サウンド発生用ＩＣを直接制御する。レジスター番号でＩＣ内部のレジスターを指定し，データを書き込む。詳細は，１７２ページ参照。

ＰＬＡＹ＜ボイスＡ文字列＞〔，＜ボイスＢ文字列＞〕〔，＜ボイスＣ文字列＞〕

音楽を発生する。３音までの和音が出せる。
文字列には，ミュージック・マクロ・ランゲージと呼ばれる音楽演奏のための記号を書く。詳細は，１７５ページ参照。

ＣＯＮＳＯＬＥ　〔＜スクロール開始行＞〕，〔＜スクロールの長さ＞〕，
〔＜ファンクションキー表示＞〕，〔＜キークリック音＞〕

画面モード１，２の場合のテキスト画面の制御をする。

スクロール開始行	０から最終行まで。最終行は，ファンクションキーを表示しているとき１４，表示していないとき１５。これより大きい値を使うと，最終行からとみなされる。
スクロールの長さ	１から最終行まで。最終行より大きいと最終行まで。
ファンクションキー表示	０なら表示しない。１なら表示する。
キークリック音	０ならキーイン音なし。１ならキーインの音あり。

ＣＯＬＯＲ　〔＜文字やグラフィックスの色＞〕〔，＜背景色＞〕
〔，＜色の組み合わせ＞〕

画面の色を指定する。詳細は，１７６ページ参照。

ＣＬＳ

アクティブページ（文字や図形が描かれているページ）を消す。

ＬＯＣＡＴＥ　Ｘ，Ｙ

２８ページ参照。カーソルを消すことはできない。

ＰＳＥＴ（Ｘ，Ｙ）〔，＜色＞〕

画面上の座標（Ｘ，Ｙ）の位置に点を描く。Ｘ，Ｙの範囲はモードによらない。
Ｘは０～２５５，Ｙは０～１９１。左上が（０，０）である。
色を指定しないとフォアグラウンドカラーになる。１７６ページ参照。

ＰＲＥＳＥＴ（Ｘ，Ｙ）

ＰＳＥＴで描いた点を消す。バックグラウンドカラーでＰＳＥＴするのと同じ。

LINE〔(A，B)〕－(C，D)〔，＜色＞〕〔，＜BまたはBF＞〕

はじめの座標を省略すると，直前に実行したLINE，PSET，PRESET，POINTで指定した座標からの意味となる。なければ（0，0）からとなる。色は，176ページ参照。B，BFはPC－8001と同じ。

PAINT(X，Y)〔，＜色＞〔，＜境界色＞〕〕

指定した点から境界色で囲まれた範囲を指定した色で塗りつぶす。色は，176ページ参照。

SCREEN 〔＜モード＞〕〔，＜アクティブページ＞〕〔，＜ビジュアルページ＞〕

画面モード	1	文字モード	
	2	64×48グラフィックス，文字	
	3	128×192グラフィックス	
	4	256×192グラフィックス	
アクティブページ（文字や図形を描く）	1	ページ1に描く	
	2	ページ2に描く	
	3	ページ3に描く	(拡張RAMが必要)
	4	ページ4に描く	(拡張RAMが必要)
ビジュアルページ（画面に表示する）	1	ページ1を表示する	
	2	ページ2を表示する	
	3	ページ3を表示する	(拡張RAMが必要)
	4	ページ4を表示する	(拡張RAMが必要)

ページ1はモード1と2が可能。ページ2，3，4はモード1，2，3，4が可能。

(3) 入出力ステートメント

INPUT＃－＜入力機器番号＞，＜変数＞〔，＜変数＞〕

入力機器番号	＃－0	キーボード
	＃－1	カセットテープレコーダー
	＃－2	RS－232C

PRINT＃－＜出力機器番号＞，＜式＞〔，＜式＞〕……

出力機器番号	＃－0	ディスプレイ
	＃－1	カセットテープレコーダー
	＃－2	RS－232C
	＃－3	プリンター

LCOPY

画面と同じ絵柄を専用プリンターに出力する。

（4）　一般関数

POINT（X，Y）

指定した位置の点の色コードが求められる。１７６ページ参照。

TIME

ＰＣ－６００１内部の時間の経過を表す数値をあたえる。数値は，（1／１０２４）秒ごとに1増える。ＰＣ－６００１内部の1秒は，（１０００／１０２４＝０．９７６５６２５）秒なので，正しい1秒にするには1．０２４を掛ける必要がある。これでもハードの影響をうけて誤差がでることに注意。

USR（X）

機械語プログラムを実行する。Ｘは機械語プログラムに渡す値。ＢＡＳＩＣ側では，機械語プログラムの開始番地を＆ＨＦＡＥＢ番地と＆ＨＦＡＥＣ番地に，ＰＯＫＥ文で下位バイト，上位バイトの順に書き込む。機械語プログラムでは，７４１Ｈ番地をコールすることにより，データＸの値をＤＥレジスターに得ることができる。Ｘの値は，－３２７６８～３２７６７の間。機械語プログラムからＢＡＳＩＣプログラムにデータを返すには，ＡＢレジスターにデータを入れてＤ１６Ｈ番地にジャンプする。

（例）任意の整数に5を足す（ＰＣ－８００１の例も参照せよ。４８ページ）

```
10  CLEAR 50, &HCFFF
20  FOR J=0 TO 11:READ M
30  POKE &HD000+J, M:NEXT J
40  DATA &HCD, &H41, &H07, &H21, &H05, &H00
50  DATA &H19, &H7C, &H45, &HC3, &H16, &H0D
60  POKE &HFAEB, &H00:POKE &HFAEC, &HD0
70  INPUT "データ は";R
80  W=USR(R)
90  PRINT W
```

機械語部分をニーモニックで書くと右のようになる。変換表参照。

```
CALL 0741H
LD   HL, 5
ADD  HL, DE
LD   A, H
LD   B, L
JP   0D16H
```

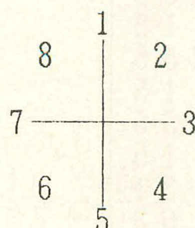
(5) 入出力関数

STICK (n)

ジョイスティックの状態をあたえる。

n＝0　キーボードカーソルキー
n＝1　ジョイスティック1
n＝2　ジョイスティック2

カーソルキーを押した状態，ジョイスティックをたおした状態は，下の数値と対応している。



カーソルキーで2・4・6・8など斜め方向を表すには2個同時にキーを押す。
たとえば，6は←キーと↓キーを同時に押す。

STRIG (n)

ジョイスティックのトリガーボタンの状態，キーボードのスペースキーの状態をあたえる。押されていれば1，はなされていれば0を示す。

n＝0　キーボードスペースキー
n＝1　ジョイスティック1ボタン
n＝2　ジョイスティック2ボタン

(例)

```
10 PRINT STRIG(1) : GOTO 10
```

ジョイスティック1のトリガーボタンが押されていれば1，はなされていれば0を表示し続ける。

(6) ストリング関数

CHR$ (X)

80ページ参照。例外に以下のものがある。

CHR$ (5)	CTRLキー＋E
CHR$ (8)	DELキー
CHR$ (12)	CLRキー
CHR$ (13)	RETURNキー
CHR$ (18)	INSキー
CHR$ (21)	CTRLキー＋U
CHR$ (34)	ダブルクォーテーション (”)

各機能の詳細

SOUND <レジスター番号>，<データ>

＊レジスターの表

レジスター	機　　　能	データの範囲	
0	ボイス1の高さ（細かく変える）	0～255	
1	ボイス1の高さ（粗く変える）	0～ 15	
2	ボイス2の高さ（細かく変える）	0～255	
3	ボイス2の高さ（粗く変える）	0～ 15	
4	ボイス3の高さ（細かく変える）	0～255	
5	ボイス3の高さ（粗く変える）	0～ 15	
6	ノイズの高さ	0～ 31	
7	ノイズ，ボイスのオン・オフ	0～ 63	
8	第1音の大きさ	0～ 15	16ならトレモロ
9	第2音の大きさ	0～ 15	16ならトレモロ
10	第3音の大きさ	0～ 15	16ならトレモロ
11	トレモロの早さ（細かく変える）	0～255	
12	トレモロの早さ（粗く変える）	0～255	
13	トレモロの形（エンベロープ）	0～ 15	

＊音を出す時は次のように考えればよい。

1．ボイスを出すか，ノイズを出すかをきめる。ボイスとは，きれいな音のこと。ノイズとは，シューあるいはザーという音のこと。混ぜることもできる。
2．ボイス，ノイズそれぞれの音の高さをきめる。
3．音の大きさをきめる。エンベロープを使うときは，16をセットする。
4．エンベロープの早さと形をきめる。

＊レジスター〔0，1〕，〔2，3〕，〔4，5〕に入れる値

それぞれ2つのレジスターを1組として，各組には各ボイスの周波数を決定する数値を入れる。数値は次の式によって計算する。

TP＝fclock／（16×fT）　CT＋FT／256＝TP／256

fclock：1．99675MHz
fT　　：出力する周波数
FT　　：レジスター0，2，4いずれかの値
CT　　：レジスター1，3，5いずれかの値

ボイス1の出力する周波数を100Hzとすると

TP＝1．99675×10^6／（16×（1×10^2））≒1248
CT＋FT／256＝1248／256＝4＋224／256

よって

FT＝２２４　CT＝４　（いずれも１０進数）

と設定すればよい。

＊レジスター６に入れる値

ノイズの周波数を決定する数値を入れる。数値は次の式によって計算する。

NP＝fclock／（16×fN）

fclock：1．99675MHz
fN　　：出力するノイズ周波数
NP　　：レジスター６に設定する数値

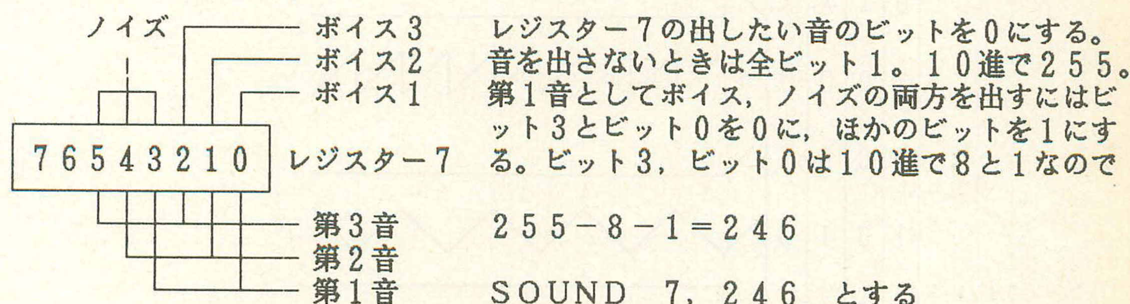
出力するノイズ周波数を１０KHzとすると

NP＝1．99675×10＾6／（16×（10×10＾3））≒12

よってレジスター６には，１２（１０進数）を設定すればよい。

＊レジスター７に入れる値

サウンド発生用ICは３つのボイスと１つのノイズをもっている。
レジスター７はこれらをオン・オフすることによって合成音をつくる。



レジスター７の出したい音のビットを０にする。音を出さないときは全ビット１。１０進で２５５。第１音としてボイス，ノイズの両方を出すにはビット３とビット０を０に，ほかのビットを１にする。ビット３，ビット０は１０進で８と１なので

２５５－８－１＝２４６

SOUND　７，２４６　とする

＊レジスター８，９，１０に入れる値

各音の大きさを決定する値を入れる。０～１５の値なら音の大きさは一定。０なら無音。１６なら音の大きさが変化するモード（エンベロープモード）となる。

＊レジスター１１，１２に入れる値

２つのレジスターにエンベロープの周期を決定する数値を入れる。数値は，次の式によって計算する。

EP＝fclock／（256×fE）　CT＋FT／256＝EP／256

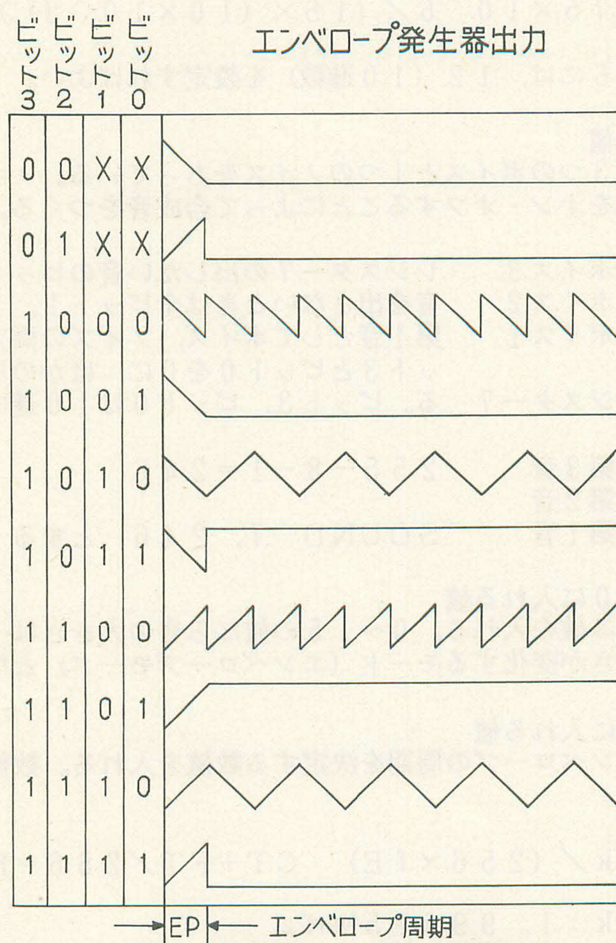
fclock：1．99675MHz
fE　　：音が増加，減少する周波数
FT　　：レジスター１１に設定する数値
CT　　：レジスター１２に設定する数値

音が減衰して０になるまでの時間を０．５秒に設定すると

ｆＥ＝１／０．５＝２Ｈｚ
ＥＰ＝１．９９６７５×１０＾６／（２５６×２）≒３９００
ＣＴ＋ＦＴ／２５６＝３９００／２５６＝１５＋６０／２５６

よってＦＴ＝６０　ＣＴ＝１５と設定する。

＊レジスター１１，１２に入れる値
エンベロープの形状を決定する値を入れる。レジスター１３のビット０～３の値が次のような波形と対応する。ｘは０でも１でも影響ない。



PLAY<ボイスA文字列>〔，<ボイスB文字列>〕〔，<ボイスC文字列>〕

ミュージック・マクロ・ランゲージ（以下，MMLと書く）の書きかた

＊音程

C	D	E	F	G	A	B
ド	レ	ミ	ファ	ソ	ラ	シ

半音は，後ろに♯なら＋を，♭なら－をつける

＊音域（オクターブ）

Ｏｎ　ｎは１～８　オクターブを変える。はじめはＯ４になっている。

＊長さ

音符の長さを数字でしめす。４分音符なら４，１６分音符なら１６。
符点つきの音符は，ピリオドを符点の数だけつける。

ＯｎＸ〔<＋または－>〕ｍ〔．〔．〕〕

- 符点　ひとつ付けると半分ずつのびる
- 長さ　４分音符なら　４
- 半音　シャープなら＋，フラットなら－
- 音程　ドなら　Ｃ
- オクターブ

＊休符

Ｒｎ　ｎは１～６４　休符。

＊各文字の意味

Ｍｘ	エンベロープ周期の設定	（ｘは１～６５５３５）	
Ｓｘ	エンベロープ形状の設定	（ｘは０～１５）	
Ｖｘ	音量の設定	（ｘは０～１５）	指定しないときは　８
Ｌｘ	長さの設定	（ｘは１～６４）	指定しないときは　４
Ｏｘ	オクターブの設定	（ｘは１～８）	指定しないときは　４
Ｔｘ	テンポの設定	（ｘは３２～２５５）	指定しないときは　１２０
Ｒｘ	休符の設定	（ｘは１～６４）	指定しないときは　４
Ｘ＋	音を半音上げる		
Ｘ－	音を半音下げる		
Ｎｘ	数値で音程を指定する	（ｘは１～９６）	

＊注意点

・Ｌでは長さを指定しない音符の長さを決める。いちばん多く使うものにしておけば，そのつど長さを指定しなくてもすむので便利。休符の長さは変わらない。
・Ｍ，ＳとＶとはどちらか一方しか指定できない。
・Ｎで音程を指定するときは，音程を表すｘと長さを表すｘとの間をセミコロンでくぎる。Ｎでだせる最低音Ｎ１は，Ｏ１Ｃ＋と同じ高さ。ｘが１増えるごとに半音ずつ上がり，９６ではＯ８Ｂと同じ高さ。

COLOR 〔<文字やグラフィックの色>〕〔, <背景色>〕
〔, <色の組み合わせ>〕

各モードでのパラメーターの組み合わせを示す。

以下の表では，各パラメーターを次の記号で表す。

文字やグラフィックの色	フォアグラウンドカラー	F
背景色	バックグラウンドカラー	B
色の組み合わせ	カラーセット	C

<モード1>

F	B	C	文字の色
1	—	1	緑
2	—	1	緑・リバース
3	—	1	だいだい
4	—	1	だいだい・リバース
1	—	2	だいだい
2	—	2	だいだい・リバース
3	—	2	緑
4	—	2	緑・リバース

<モード2>

F	B	C	文字の色	グラフィックスの色
0	0～8	1	緑	黒
1	0～8	1	緑	緑
2	0～8	1	緑・リバース	黄
3	0～8	1	だいだい	青
4	0～8	1	だいだい・リバース	赤
5	0～8	1	だいだい・リバース	白
6	0～8	1	だいだい・リバース	水色（シアン）
7	0～8	1	だいだい・リバース	紫（マゼンタ）
8	0～8	1	だいだい・リバース	だいだい
0	0～8	2	だいだい	黒
1	0～8	2	だいだい	白
2	0～8	2	だいだい・リバース	水色（シアン）
3	0～8	2	緑	紫（マゼンタ）
4	0～8	2	緑・リバース	だいだい
5	0～8	2	緑・リバース	緑
6	0～8	2	緑・リバース	黄
7	0～8	2	緑・リバース	青
8	0～8	2	緑・リバース	赤

＜モード3＞

F	B	C	文字またはグラフィックスの色
1	1～4	1	緑
2	1～4	1	黄
3	1～4	1	青
4	1～4	1	赤
1	1～4	2	白
2	1～4	2	水色（シアン）
3	1～4	2	紫（マゼンタ）
4	1～4	2	だいだい

＜モード4＞

F	B	C	文字またはグラフィックスの色
1	0	1	黒地に緑
0	1	1	緑地に黒
1	0	2	黒地に白
0	1	2	白地に黒

SCREEN 2,2,2:COLOR 0,0,1
SCREEN 4,2,2:COLOR 1,0,2

PC—6001キャラクター・コード表

16進数は横が上位4ビット縦が下位4ビットマスの中の数字は16進数に対応する10進数

たとえば4BはKだが10進では75になる

16進	0	1	2	3	4	5	6	7
0	0	π 16	32	0 48	@ 64	P 80	96	p 112
1	月 1	┴ 17	! 33	1 49	A 65	Q 81	a 97	q 113
2	火 2	┬ 18	" 34	2 50	B 66	R 82	b 98	r 114
3	水 3	┤ 19	# 35	3 51	C 67	S 83	c 99	s 115
4	木 4	├ 20	$ 36	4 52	D 68	T 84	d 100	t 116
5	金 5	┼ 21	% 37	5 53	E 69	U 85	e 101	u 117
6	土 6	│ 22	& 38	6 54	F 70	V 86	f 102	v 118
7	日 7	─ 23	' 39	7 55	G 71	W 87	g 103	w 119
8	年 8	┌ 24	(40	8 56	H 72	X 88	h 104	x 120
9	円 9	┐ 25	) 41	9 57	I 73	Y 89	i 105	y 121
A	時 10	└ 26	* 42	: 58	J 74	Z 90	j 106	z 122
B	分 11	┘ 27	+ 43	; 59	K 75	[91	k 107	{ 123
C	秒 12	╳ 28	, 44	< 60	L 76	¥ 92	l 108	¦ 124
D	百 13	大 29	— 45	= 61	M 77	] 93	m 109	} 125
E	千 14	中 30	. 46	> 62	N 78	^ 94	n 110	~ 126
F	万 15	小 31	/ 47	? 63	O 79	_ 95	o 111	127

8	9	A	B	C	D	E	F	16進
♠ 128	144	160	ー 176	タ 192	ミ 208	た 224	み 240	0
♥ 129	あ 145	。 161	ア 177	チ 193	ム 209	ち 225	む 241	1
♣ 130	い 146	「 162	イ 178	ツ 194	メ 210	つ 226	め 242	2
♦ 131	う 147	」 163	ウ 179	テ 195	モ 211	て 227	も 243	3
○ 132	え 148	、 164	エ 180	ト 196	ヤ 212	と 228	や 244	4
● 133	お 149	・ 165	オ 181	ナ 197	ユ 213	な 229	ゆ 245	5
を 134	か 150	ヲ 166	カ 182	ニ 198	ヨ 214	に 230	よ 246	6
ぁ 135	き 151	ァ 167	キ 183	ヌ 199	ラ 215	ぬ 231	ら 247	7
ぃ 136	く 152	ィ 168	ク 184	ネ 200	リ 216	ね 232	り 248	8
ぅ 137	け 153	ゥ 169	ケ 185	ノ 201	ル 217	の 233	る 249	9
ぇ 138	こ 154	ェ 170	コ 186	ハ 202	レ 218	は 234	れ 250	A
ぉ 139	さ 155	ォ 171	サ 187	ヒ 203	ロ 219	ひ 235	ろ 251	B
ゃ 140	し 156	ャ 172	シ 188	フ 204	ワ 220	ふ 236	わ 252	C
ゅ 141	す 157	ュ 173	ス 189	ヘ 205	ン 221	へ 237	ん 253	D
ょ 142	せ 158	ョ 174	セ 190	ホ 206	゛ 222	ほ 238	254	E
っ 143	そ 159	ッ 175	ソ 191	マ 207	゜ 223	ま 239	255	F

ＰＣ－６００１　メモリーアドレスマップ

ＲＡＭ１６Ｋバイトの場合
（スクリーンページが２ページ）

16進番地	内容	備考
FFFF–EA00	ＢＡＳＩＣ　ＳＹＳＴＥＭ	
EA00–E000	ページ　２	
E000–C400	プログラム	
C400–C000	ページ　１	
C000–8000		
8000–7000		拡張ＲＯＭエリア
7000–6000	キャラクタージェネレーター（バンク切り替え）	拡張ＲＯＭエリア
6000–4000		拡張ＲＯＭエリア
4000–0000	ＢＡＳＩＣ インタープリター領域	

ＲＡＭ３２Ｋバイトの場合
（スクリーンページが４ページ）

16進番地	内容	備考
FFFF–EA00	ＢＡＳＩＣ　ＳＹＳＴＥＭ	
EA00–E000	ページ　２	
E000–C000	ページ　３	
C000–A000	ページ　４	拡張ＲＡＭエリア
A000–8400	プログラム	拡張ＲＡＭエリア
8400–8000	ページ　１	拡張ＲＡＭエリア
8000–7000		拡張ＲＯＭエリア
7000–6000	キャラクタージェネレーター（バンク切り替え）	拡張ＲＯＭエリア
6000–4000		拡張ＲＯＭエリア
4000–0000	ＢＡＳＩＣ インタープリター領域	

ＲＡＭを１６Ｋバイト拡張する場合
　　ＲＯＭ＆ＲＡＭカートリッジを接続する。

ＲＯＭを１６Ｋバイト拡張する場合
　　ＲＯＭ＆ＲＡＭカートリッジまたはＲＯＭカートリッジを接続する。

PC−6001 エラーメッセージ

?BS Error (Bad subscript Error)
配列の添字がDIMで宣言したものより大きい。
添字の個数が正しくない。

?CN Error (Continue Error)
CONT命令を実行できない。プログラムを変更したあとでCONT命令は実行できない。

?DD Error (Dupulicate Definition Error)
同じ名前の配列をもう1度定義した。

?FC Error (Function Call Error)
許容範囲外の数字を使った。

?ID Error (Illegal Direct Error)
ダイレクトモードでは使えないコマンドを使った。
INPUT文，DEFFN文など。

?LS Error (Long String Error)
文字列が長すぎる。255文字が限度。

?MO Error (Missing Operand Error)
不完全な記述がある。代入命令の右辺がないなど。

?NF Error (Next without For Error)
FORとNEXTが対応していない。NEXTがあるのにFORがない。

?OD Error (Out of Data Error)
READ命令を実行したが，DATAがない。

?OM Error (Out of Memory Error)
プログラムが大きすぎてメモリー不足になった。
サブルーチンの中で自分自身のサブルーチンを呼んだ。

?OS Error (Out of String Space Error)
文字列用の領域が不足。CLEAR文で増やすこと。

?OV Error (Over flow Error)
計算結果が許容範囲を越えた。大きすぎるか，小さすぎる。

?RG Error (Return without Gosub Error)
GOSUB命令がないのにRETURNがある。

ＰＣ－６００１ エラーメッセージ

?SN Error (Syntax Error)
文法違反。

?ST Error (String too Complex Error)
文字式が複雑すぎる。

?TM Error (Type Mismatch Error)
文字や数値の扱いかたが正しくない。数値変数に文字を入れるなど。

?TR Error (Tape Read Error)
テープが正しく読み込めない。

?UF Error (Undefined Function Error)
ＤＥＦ ＦＮ で定義されていないユーザー関数を呼んだ。

?UL Error (Undefined Line number Error)
指定した行番号がない。

?/0 Error (Division by Zero Error)
０で割り算している。

PC－6001シリーズ　ラインアップ

型　　名	品　　名	特　　長
PC－6001	PC－6000シリーズのパーソナルコンピューター	
PC－6005	ROMカートリッジ	2K，4K，8KバイトのマスクROM，8KバイトのPROMを使うことができる
PC－6006	ROM&RAMカートリッジ	RAM16Kバイト ROMは上と同じ
PC－6021	40桁専用サーマルプリンター	画面プリントを取ることができる
PC－6031	ミニフロッピーディスクユニット	
PC－6041	12インチ・グリーンディスプレイ	
PC－6042	12インチ・カラーディスプレイ	
PC－6051	タッチパネル（ディジタイザー）	パネル上の指定した点の位置を高精度で読み取ることができる
PC－6061	RS－232Cボード	300，600，1200，2400，4800ボーの選択が可能
PC－6071	フィーダー切り替えスイッチ	
PC－6072	キーボード・オーバーレイシート	無地
PC－6073	同上	ひらがな用
PC－6081	データレコーダー	パーソナルコンピューター用カセットテープレコーダー
PC－6082	データレコーダー	同上。プログラムの頭出しが可能

<＜索　　引＞

＜索　　引＞

＜索　　　引＞

＜索　　　引＞

＜索　　引＞

＜索　　引＞

＜索　　　引＞

パソコンPCシリーズ　ハンドブック

昭和57年3月　1日　第1刷　　定価800円
昭和57年6月25日　第7刷
編者　　朝日新聞社電子計算室
発行者　　波多野公介
印刷製本　凸版印刷株式会社
発行所　　朝日新聞社
〒104　東京都中央区築地5－3－2
電話　03（545）0131（代表）
編集＝出版局プロジェクト室　　販売＝出版販売部
振替　東京0－1730

K. Mori

パソコンPCシリーズ 8001 6001 ハンドブック

●本書の主な内容

 2055-258141-0042　Printed in Japan

定価800円